KI-Modelle in den Sozialwissenschaften: logische Struktur und wissensbasierte Systeme von Balancetheorien

Manhart, Klaus

Veröffentlichungsversion / Published Version
Monographie / monograph

Klaus Manhart · KI-Modelle in den Sozialwissenschaften

Scientia Nova

Herausgegeben von
Rainer Hegselmann, Gebhard Kirchgässner,
Hans Lenk, Siegwart Lindenberg,
Werner Raub, Thomas Voss

Bisher erschienen u. a.:
Robert Axelrod, Die Evolution der Kooperation
Karl H. Borch, Wirtschaftliches Verhalten bei Unsicherheit
Churchmann/Ackoff/Arnoff, Operations Research
James S. Coleman, Grundlagen der Sozialtheorie, 3 Bände
Erklären und Verstehen in der Wissenschaft
Evolution und Spieltheorie
Bruno de Finetti, Wahrscheinlichkeitstheorie
Robert Frank, Strategie der Emotionen
Richard C. Jeffrey, Logik der Entscheidungen
Peter Kappelhoff, Soziale Tauschsysteme
Moralische Entscheidung und rationale Wahl
Bernd Lahno, Versprechen. Überlegungen zu einer künstlichen Tugend
Nagel/Newman, Der Gödelsche Beweis
John von Neumann, Die Rechenmaschine und das Gehirn
Julian Nida-Rümelin, Kritik des Konsequentialismus
Erhard Oeser, Wissenschaft und Information
Howard Raiffa, Einführung in die Entscheidungstheorie
Erwin Schrödinger, Was ist ein Naturgesetz?
Rudolf Schüßler, Kooperation unter Egoisten: vier Dilemmata
Thomas Voss, Rationale Akteure und soziale Institutionen
Hermann Weyl, Philosophie der Mathematik und Naturwissenschaft

Klaus Manhart

KI-Modelle in den Sozialwissenschaften

Logische Struktur und wissensbasierte Systeme von Balancetheorien

R. Oldenbourg Verlag München Wien 1995

Gedruckt mit Unterstützung des Förderungs- und
Beihilfefonds Wissenschaft der VG Wort

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Manhart, Klaus:
KI-Modelle in den Sozialwissenschaften : logische Struktur und
wissensbasierte Systeme von Balancetheorien / Klaus Manhart.
- München ; Wien : Oldenbourg, 1995
(Scientia nova)
ISBN 3–486–56105–7

Umschlaggestaltung: Dieter Vollendorf
Druck und Bindung: WB-Druck, Rieden am Forggensee

ISBN 3–486–56105–7

Inhalt

Vorwort

Diese Arbeit befaßt sich mit der Anwendung nicht-numerischer, "wissensbasierter" Verfahren der Informatik/Künstlichen Intelligenz (KI) auf die Modellierung sozialwissenschaftlicher Theorien. Das Ziel besteht darin zu zeigen, daß der Einsatz von KI-Programmen eine fruchtbare methodische Erweiterung für die Sozialwissenschaften darstellt. Die wissenbasierte Modellierung erfolgt auf dem Hintergrund logisch rekonstruierter Theorien im Rahmen eines neueren wissenschaftstheoretischen Ansatzes: der strukturalistischen Theorienkonzeption.

Das Buch wendet sich an drei Gruppen von Interessenten. An erster Stelle zu nennen sind Sozialwissenschaftler (Psychologen und Soziologen) mit Interesse an Computersimulation, Modellbildung und der Präzisierung sozialwissenschaftlicher Theorien. Eine zweite Interessensgruppe sind Wissenschaftstheoretiker bzw. Philosophen, die offen sind für die metatheoretische Behandlung nicht-naturwissenschaftlicher Theorien. Schließlich richtet sich dieses Buch auch an die Experten aus der KI und Informatik. Zwar tragen die hier entwickelten Ideen nichts Wesentliches zu einer Erweiterung des Fachwissens in diesen Gebieten bei, aber sie erschließen einen neuen Anwendungsbereich jenseits der traditionellen Verbindung zwischen KI und kognitiver Psychologie. Die Arbeit kann somit auch für KI-Wissenschaftler und Informatiker anregend sein, die sich mit anderen als den sonst üblichen, praxisorientierten oder kognitionswissenschaftlichen Anwendungen anfreunden können.

Das vorliegende Buch ist die überarbeitete Fassung einer Dissertation, die im Sommer 1993 am Institut für Philosophie, Logik und Wissenschaftstheorie der Ludwig-Maximilians-Universität München vorgelegt wurde. Ich danke allen, die mich direkt oder indirekt bei der Vorbereitung, Abfassung und Überarbeitung des Manuskripts unterstützt haben. Explizit erwähnen möchte ich meinen Betreuer, Herrn Prof. Dr. Wolfgang Balzer, der mir in allen Phasen wertvolle und entscheidende Anregung und Hilfestellung gab. Ich verdanke ihm die logisch saubere Rekonstruktion der Theorien ebenso wie wesentliche Ideen für die Balancegenerierungs-Verfahren. Dank für die Unterstützung und Förderung meines Vorhabens schulde ich ebenfalls Herrn Prof. Dr. Rolf Ziegler, an dessen Lehrstuhl ich während der Abfassung dieser Arbeit beschäftigt war. Wichtige Literaturhinweise verdanke ich Herrn Prof. Dr. Thomas Voss und zwei anonymen Gutachtern der "Zeitschrift für Sozialpsychologie". Herr Voss und Frau Nicole Saam haben sich freundlicherweise bereit erklärt, das Manuskript vor der Drucklegung noch einmal zu lesen. Gedankt sei schließlich auch noch dem Förderungs- und Beihilfefonds Wissenschaft der VG Wort für den von ihr gewährten Druckkostenzuschuß. Gewidmet sei dieses Buch dem Gedenken an Herrn Prof. Dr. Wolfgang Stegmüller, der mein Interesse für Wissenschaftstheorie weckte und dessen Lehrveranstaltungen ich nie vergessen werde.

München, im September 1994 Klaus Manhart

Einleitung

Die Entwicklung von Theorien kann als das eigentliche Ziel empirischer Wissenschaft betrachtet werden, und der Computer kann dabei als wertvolles unterstützendes Instrument eingesetzt werden. Zum einen lassen sich bestehende Theorien in Form von Computerprogrammen formal präzise modellieren und weiterentwickeln, zum anderen können durch die Computermodellierung empirischer Daten völlig neue Theorien und Erklärungsmöglichkeiten entstehen. Substanzwissenschaftler, die Computersimulation einsetzen, betonen immer wieder, daß dadurch der Gang der Theoriebildung wesentlich beschleunigt und erleichtert wird. Neben dem traditionellen Experiment werden Rechnermodelle deshalb mehr und mehr als zweite Säule der empirischen Wissenschaften und als Schlüsseltechnologie für die nächsten Jahrzehnte betrachtet.

Dem mächtigen theoretischen Potential des Computers steht die faktische Verwendungsweise in den Substanzwissenschaften gegenüber. In den Substanzwissenschaften wird der Computer weitgehend als "Number-Cruncher" angesehen, der lediglich arithmetische Operationen mit Zahlen ausführen kann. Insbesondere in den Human- und Sozialwissenschaften ergibt sich dadurch ein Dilemma, da die meisten Theorien in Umgangssprache vorliegen und sich selten in die Kalküle und Numerikmodelle übersetzen lassen, mit denen normalerweise Naturwissenschaftler arbeiten. Dieses Dilemma verschwindet jedoch, wenn man den Computer als universelle Symbolverarbeitungsmaschine betrachtet, die beliebige Operationen über beliebige Zeichen ausführen und "Wissen" verarbeiten kann.

Versteht man den Computer als Maschine, die auch qualitative, nicht-numerische Informationen bearbeiten kann, so bewegt man sich zwangsläufig in Gefilden der Künstlichen Intelligenz (KI). Diese Grenzdisziplin zwischen Informatik und Humanwissenschaft beschäftigt sich unter anderem mit der maschinellen Repräsentation und Bearbeitung nicht-numerischen, symbolischen Wissens. Da empirische Theorien vielfach nicht-numerische Strukturen enthalten, die als eine Art Expertenwissen betrachtet werden können, liegt es auf der Hand, das Potential der KI für die empirische Modellbildung zu nutzen und einzusetzen.

In der vorliegenden Arbeit wird versucht, bestimmte Konzepte und Verfahren der KI auf die Modellierung empirischer Theorien anzuwenden. Im Mittelpunkt steht dabei die symbolische Repräsentation in Form von wissens- oder regelbasierten Systemen, bei der einfache Techniken, wie sie bei Expertensystemen üblich sind, genutzt werden. Die theoretischen Strukturen werden Schritt für Schritt in das Computerprogramm übersetzt. Die Implementation ist dabei nicht Selbstzweck, sondern wird eingebettet in einen substanzwissenschaftlichen und wissenschaftstheoretischen Rahmen. Sowohl theoretische als auch metatheoretische Aspekte der implementierten Modelle werden dargelegt und diskutiert. Neben der bloßen Repräsentation ist ein wichtiges angestrebtes Ziel die Entdeckung substanzwissenschaftlich interessanter theoretischer Einsichten und Konsequenzen.

Die substanzwissenschaftliche Perspektive ist die der Sozialwissenschaften, die allgemeinen Ausführungen dürften aber auch auf andere empirische Wissenschaften zutreffen. Die behandelten Modellierungsobjekte - Gleichgewichts- oder Balancetheorien - entstammen den Sozialwissenschaften und liegen im Grenzbereich zwischen Psychologie und Soziologie. Dieser Theorienstrang war ursprünglich rein psychologisch orientiert und wurde im Lauf der Zeit auf die Analyse sozialer Netze ausgedehnt. Die theoretische Entwicklung ist relativ überschaubar und abgeschlossen. Die wichtigsten Entwicklungsschritte der Balancetheorien werden im zweiten Teil dargestellt und logisch mit den Mitteln des strukturalistischen Theorienkonzepts rekonstruiert. Diese in der analytischen Wissenschaftstheorie ausgearbeitete neue Interpretation empirischer Theorien bietet einen einheitlichen metatheoretischen Rahmen und weist bestimmte Parallelen zu dem wissensbasierten Ansatz auf. Zu der sozialwissenschaftlichen Perspektive kommt also noch der Blickwinkel des Wissenschaftsphilosophen hinzu und wir werden - je nach Kontext - einmal die Position des Substanzwissenschaftlers und das andere Mal eher die des Wissenschaftsphilosophen einnehmen.

Die Thematik dieser Arbeit liegt somit im Schnittpunkt von drei verschiedenen Disziplinen. Die behandelten Theorien entstammen inhaltlich den Sozialwissenschaften (Soziologie und Sozialpsychologie). Die Theorien werden wissenschaftstheoretisch mit dem Instrumentarium des strukturalistischen Theorienkonzepts dargestellt und interpretiert. Bei der Modellierung verwenden wir einfache Verfahren und Techniken aus der KI. Dabei interessieren besonders die Möglichkeiten und das Potential von Wissenschaftstheorie in Verbindung mit KI für die Deutung, Präzisierung und Modellierung sozialwissenschaftlicher Theorien.

Im ersten Teil wird versucht, einen Rahmen zu schaffen, in den die nachfolgenden wissensbasierten Modelle eingeordnet und in dem sie interpretiert werden können.

Kapitel 1 rekonstruiert in den Sozialwissenschaften übliche Theorie- und Modellbegriffe, die wir in Beziehung setzen zu den entsprechenden Konzepten der Wissenschaftstheorie. In Kapitel 2 werden grundlegende Aspekte der Computermodellierung vorgestellt, Nutzen von Computermodellen und Simulationstypen in den Sozialwissenschaften. In diesem Zusammenhang gehen wir auf die Probleme ein, die mit den gängigen Modellierungsformen in den Sozialwissenschaften verknüpft sind. In Kapitel 3 behandeln wir die in der KI- und Simulationsliteratur implizit und explizit vertretene These, nach welcher (manche) Computerprogramme direkt als Theorien zu betrachten sind. Diese These findet sich insbesondere in den Kognitionswissenschaften. Das 4. Kapitel stellt die für unsere Belange wichtigsten Aspekte der KI-basierten Wissensverarbeitung vor, wobei wir uns weitgehend auf Prädikatenlogik und Expertensystemtechniken beschränken. In dem Zusammenhang werden wir auf die hier verwendete Logik-Programmiersprache PROLOG eingehen. In Kapitel 5 stellen wir in Grundzügen das strukturalistische Theorienkonzept vor, in dessen metatheoretischem Licht wir die nachfolgenden Theorien betrachten werden.

Im zweiten Teil werden die allgemeinen Aussagen des ersten Teils auf konkrete Theorien angewendet. Das Fundament bildet dabei die Beschreibung eines balancetheoretischen Theorienstrangs in seiner historischen Entwicklung. Zunächst wird im

6. Kapitel die einfache Ausgangstheorie von Heider informell dargestellt, strukturalistisch interpretiert und anschließend in zwei Versionen in ein Computerprogramm umgesetzt: in ein Datengenerierungs-Modell und in ein Modell, in welchem strukturalistische Prädikate definiert werden. In Kapitel 7 stellen wir eine Verallgemeinerung der Heider-Theorie von Abelson/Rosenberg vor und implementieren sie als Datengenerierungs-Modell. Kapitel 8 stellt zunächst die weitere theoretische Entwicklung der Balancetheorien vor, in der vor allem graphentheoretische Aspekte relevant werden. Die hier und im folgenden Kapitel beschriebenen Modelle haben eine wesentlich größere Zahl von Anwendungen und lassen sich insbesondere auf soziale Netze beziehen. Das allgemeinste Modell des transitiven Graphen wird in Kapitel 9 behandelt. Mit dieser - im Vergleich zu den anderen Modellen - etwas umfangreicheren, abschließenden Implementierung versuchen wir, theoretisch interessante Konsequenzen aus dem Modell herzuleiten. Hierzu werden in Kapitel 10 Generierungshypothesen und ihre theoretischen Folgerungen vorgestellt. Kapitel 11 diskutiert mögliche Modifikationen der Balancetheorien und des Computerprogramms. Das inhaltlich abschließende Kapitel 12 untersucht den wissenschaftstheoretischen Status der dargestellten Balancetheorien, und zwar intertheoretische Relationen und Evolution der Gleichgewichtstheorien. Im letzten Kapitel soll dann eine Zusammenfassung der wichtigsten Ergebnisse erfolgen.

Die gesamte Arbeit hindurch, insbesondere aber im zweiten Teil, wird von logischer und mengentheoretischer Notation Gebrauch gemacht. Die verwendeten Sprachmittel gehen kaum über Schulkenntnisse hinaus und werden von elementaren Textbüchern wie Halmos (1976), Tarski (1977) oder Kapitel 1 aus Stegmüller/Varga von Kibéd (1984) abgedeckt. Die wichtigsten verwendeten Symbole sind im Anhang aufgelistet.

Teil I

Grundlagen: Computermodellierung, Wissensverarbeitung, Wissenschaftstheorie

1. Theorie, Modell und Formalisierung

Theorien und Modelle sind die wesentlichen Konzepte dieser Untersuchung. Wir sollten also erst einmal festlegen, was wir darunter verstehen wollen. Experten für Fragen nach Aufbau und Struktur wissenschaftlicher Theorien und Modelle sind Wissenschaftstheoretiker oder Wissenschaftsphilosophen.[1] Sofern sich Substanzwissenschaftler überhaupt mit solchen Problemstellungen beschäftigen, lehnen sie sich meist an vorgegebene Konzepte der Wissenschaftsphilosophie an. Wir möchten im folgenden ein solches weitverbreitetes Theorienkonzept in der Rezeption von Sozialwissenschaftlern vorstellen, zunächst aber kurz die allgemeine Rolle der Wissenschaftstheorie anreißen.

Erkenntnis- und Untersuchungsobjekte der Wissenschaftstheorie sind Theorien, mit denen in den einzelnen Wissenschaften gearbeitet wird. Für den Wissenschaftstheoretiker sind Theorien aus den Substanzwissenschaften wie Mathematik, Physik, Psychologie oder Soziologie in gleicher Weise Forschungsobjekte wie für einen Physiker Naturereignisse oder für einen Soziologen soziale Phänomene. Ähnlich wie der Erfahrungswissenschaftler also Theorien verwendet, um eine Menge von Beobachtungen zu beschreiben, vorherzusagen und zu erklären, kann der Wissenschaftstheoretiker Theorien benutzen, um die verschiedenen empirischen Theorien der Einzelwissenschaften in ihrem Aufbau und in ihrer Funktion zu behandeln (Westermann 1987: 4). Die von den Wissenschaftstheoretikern entwickelten "Theorien über Theorien" werden als *Metatheorien* bezeichnet, und die Wissenschaftstheorie selbst ist eine *Metawissenschaft* zu den Substanzwissenschaften (Stegmüller 1973: 1, Balzer 1982: 1). Der Objektbereich einer wissenschaftstheoretischen Metatheorie umfaßt im Prinzip alle bisher aufgestellten empirischen Theorien. Ziel der Wissenschaftstheorie ist es, über diesen Bereich allgemeine - metatheoretische - Aussagen zu machen (Balzer 1982: 2) und damit den Aufbau und die Struktur von Theorien präziser und klarer zu machen.

Das eben genannte Ziel, allgemeine metatheoretische Aussagen über bestehende Theorien zu machen, kann man als deskriptives Vorgehen bezeichnen. Wissenschaftstheorie *deskriptiv* verstanden, *beschreibt* die Struktur und das Begriffsgerüst *faktisch*

[1] Die Ausdrücke "Wissenschaftstheorie" und "Wissenschaftsphilosophie" werden in der deutschsprachigen Literatur als Synonyme verwendet.

vorliegender Theorien. Eine dieser Position entgegengesetzte Vorstellung ist, Wissenschaftstheorie normativ zu betreiben. Wissenschaftstheorie *normativ* verstanden, versucht *begründbare Normen für korrektes wissenschaftliches Arbeiten* zu erstellen (Stegmüller 1973: 8). Die Geschichte der Wissenschaftstheorie kann durch ein ständiges Pendeln zwischen eher normativ und eher deskriptiv ausgerichteten Positionen gekennzeichnet werden.

So wie es in den Substanzwissenschaften unterschiedliche Theorien zum gleichen Objekt gibt, gibt es in der Wissenschaftsphilosophie auch verschiedene Grundpositionen, die empiristische, rationalistische, realistische oder auch idealistische Züge tragen können. Die verschiedenen wissenschaftstheoretischen Ansätze können dabei als unterschiedliche Metatheorien zu gleichen oder ähnlichen Gegenstandsbereichen gekennzeichnet werden (Westermann 1987: 3-4). Bei Wissenschaftsphilosophen und wissenschaftstheoretisch interessierten Substanzwissenschaftlern hat in den letzten Jahren ein neuer Ansatz Aufmerksamkeit gefunden, der als strukturalistische Theorienkonzeption bezeichnet wird und auf Sneed (1971) zurückgeht. Dieses deskriptiv orientierte Konzept liefert eine neuartige und fruchtbare Explikation des Theorie- *und* Modellbegriffs, und wir wollen es später als metatheoretischen Rahmen benutzen. Wir behandeln vorläufig aber den Theorie- und Modellbegriff konventionell, wie er in weiten Teilen der Sozialwissenschaften und in der Computersimulationsliteratur verwendet wird. Die Begriffe werden später in Relation zu ihrer strukturalistischen Interpretation gebracht.

1.1 Standardtheorienkonzept

Empirisch orientierte Sozialwissenschaftler vertreten im allgemeinen einen Theoriebegriff, der sich an Theorienkonzepte der analytischen Wissenschaftstheorie anlehnt. Für den strikt *empirisch* orientierten Sozialwissenschaftler ist *Beschreibung* und *Erklärung* sozialer Phänomene mit "empirisch gehaltvollen Theorien" das Ziel. Zwischen Theorien aus den Sozial- oder Geisteswissenschaften einerseits und Theorien aus den Naturwissenschaften andererseits besteht dabei *kein grundsätzlicher Unterschied* in Aufbau, Ziel und verwendeter Sprache.[2]

Wenn wir uns der Frage, über welche Metatheorie empirisch orientierte Sozialwissenschaftler verfügen, quasi aus der Vogelperspektive - also metaempirisch - nähern und uns willkürlich einige Lehrbücher und Aufsätze zur Methodologie und

[2] Diese Formulierung täuscht darüber hinweg, daß in den Sozialwissenschaften seit Beginn ihres Entstehens kontrovers und heftigst über ihre Grundlagen und Theoriebildung diskutiert wird. Während in der Physik Newtons Gravitationstheorie von faktisch allen Fachwissenschaftlern als Theorie angenommen und innerhalb der Grenzen der klassischen Mechanik anerkannt ist, gibt es in den Sozialwissenschaften weder einen Konsens, was eine Theorie ausmacht, noch gibt es Theorien, die allgemein akzeptiert sind. In ihrer Einleitung zu "Probleme der Modellierung sozialer Prozesse" beklagen Esser/Troitzsch (1991b) beispielsweise: "In kaum einem anderen Fach gehen die Meinungen darüber, was eigentlich die Eigenschaften einer akzeptablen 'Theorie' seien, so weit auseinander wie in den Sozialwissenschaften. Eine Konvergenz in den Auffassungen (und damit in den Produkten) ist auch in den aktuelleren Entwicklungen nicht festzustellen" (Esser/Troitzsch 1991b: 13).

Wissenschaftstheorie der empirischen Sozialwissenschaften herausgreifen, finden sich z.B. folgende Interpretationen des Theoriebegriffs:

- Eine Theorie ist eine "Aussagenmenge, die aus einer Reihe von Gesetzen, deskriptiven Aussagen und logischen Ableitungen besteht" (Giesen/Schmid 1977: 268).
- Eine Theorie ist ein "System von Aussagen, das mehrere Hypothesen oder Gesetze umfaßt" (Schnell/Hill/Esser 1993: 43).
- Theorien sind Satzsysteme, "die als Explanans in deduktiv-nomologischen Erklärungen verwendet werden können" (Schnell 1990: 123).
- Eine Theorie ist eine Menge von Gesetzen, wenn diese durch logische Ableitbarkeitsbeziehungen miteinander verbunden sind (Opp 1976: 78).

Alle Definitionsversuche betonen unterschiedliche Aspekte. In ihrer semantischen Schnittmenge ist ihnen aber gemeinsam, daß Theorien als *Mengen von Aussagen oder Sätzen mit nomologischem* (also *Gesetzes-*) *Charakter* betrachtet werden. Dies ist eine zentrale Feststellung, auf die wir uns im folgenden beziehen werden.

Unter Verweis auf die Tatsache, daß es in der Wissenschaftstheorie bislang weder gelungen ist, den Gesetzesbegriff in befriedigender Weise zu definieren, noch Kriterien für die Gesetzesartigkeit von Aussagen anzugeben, wird in manchen Lehrbüchern ein Gesetzesbegriff informell-pragmatisch eingeführt, wie etwa in Opp (1976: 75):

"Ein Gesetz heißt eine empirische Aussage, die 1. ohne raum-zeitlichen Bezug ist, in der 2. allen Elementen (mindestens) einer unendlichen Menge von Objekten (mindestens) ein Merkmal zugeschrieben wird, die 3. als Wenn-dann oder Je-desto-Aussage formuliert werden kann und die 4. sich empirisch relativ gut bewährt hat."

Aussagen, die lediglich die Definitionsmerkmale 1 bis 3 erfüllen, nennt Opp *gesetzesartige* Aussagen. Unter der Voraussetzung, daß die 4. Bedingung erfüllt ist, bilden dann die folgenden beiden Sätze bereits eine Theorie im Sinn von Opp (1976: 78):

"Je isolierter Personen sind, desto häufiger brechen sie Normen."

"Je häufiger Personen Normen brechen, desto eher wählen sie rechts- oder linksextreme Parteien."

"Hauptaufgabe einer Theorie, ob in der Soziologie oder in den Naturwissenschaften, ist die Erklärung oder die Unterstützung von Erklärungen" (Cohen 1980: 592). Empirische Sozialwissenschaftler berufen sich dabei meist auf das klassische Subsumtionsmodell der Erklärung (auch: covering law, deduktiv-nomologisches (D-N) oder Hempel-Oppenheim (H-O) Schema). Im D-N-Modell werden empirische Phänomene erklärt unter Rückgriff auf Gesetze; es hat im einfachsten Fall die folgende Form:

Explanans	F(a)
	Für alle x: wenn F(x) dann G(x)
Explanandum	G(a)

In diesem Schema soll "F(a)" bzw. "G(a)" den Sachverhalt ausdrücken, daß irgendein Objekt a die Eigenschaft F bzw. G hat. "G(a)" ist der zu erklärende Sachverhalt

(das *Explanandum*), "F(a)" die singuläre Randbedingung und "für alle x: wenn F(x) dann G(x)" ein einfaches Gesetz. Randbedingung und Gesetz werden zusammen als *Explanans* bezeichnet. Im deduktiv-nomologischen Erklärungsmodell wird ein Ereignis erklärt, indem es aus dem *Explanans* logisch deduziert wird (vgl. z.B. Giesen/Schmid 1977: 54ff.). G(a) wird also dadurch erklärt, daß Randbedingungen und Gesetzmäßigkeiten gesucht werden, die es erlauben, G(a) logisch abzuleiten. Sind umgekehrt Randbedingungen und Gesetze bekannt, kann G(a) prognostiziert werden. Im H-O-Schema sind Erklärung und Prognose strukturell gleichartig.

Sowohl die Problematik sozialwissenschaftlicher Gesetze als auch die des H-O-Schemas insbesondere im Zusammenhang mit statistischen Gesetzen (induktiv-statistische Erklärung) und sog. unvollständigen Erklärungen werden in den Lehrbüchern ausführlich diskutiert (vgl. Giesen/Schmid 1977, Opp 1976, Prim/Tilmann 1989). Auf diese Probleme kann hier nicht eingegangen werden.

Die eben angerissenen Deutungen von "Theorie", "Gesetz" und "Erklärung" entspringen weitgehend dem Einfluß der analytischen Wissenschaftstheorie.[3] Die Auffassung, nach der Theorien bestimmte Systeme oder Klassen von (gesetzesartigen) Sätzen sind, wird in der neueren Wissenschaftsphilosophie als *Aussagenkonzept* oder - wegen ihres starken und zeitlich andauernden Einflusses - auch als *Standardtheorienkonzept* bezeichnet. Dieses Konzept geht auf den Logischen Empirismus des Wiener Kreises um Rudolf Carnap zurück.

Der Logische Empirismus betonte die Einheit der Wissenschaft und verfolgte das Ziel, die Struktur oder den inneren Aufbau von wissenschaftlichen Theorien mit den Mitteln der modernen Logik zu analysieren. Vereinfacht gesagt, sieht der Logische Empirismus Theorien als eine Menge von Aussagen, die induktiv aus Daten und grundlegenden Fakten gewonnen wird. Das reduktionistisch-empiristische Programm von Carnap und seinen Schülern bestand zunächst darin, alles Wissen auf Sinneserfahrungen zu gründen. Alle nichtlogischen Begriffe sollten entweder durch Definitionen oder durch sog. Reduktionssätze auf Grundbegriffe mit unmittelbarem empirischem Gehalt zurückgeführt werden. Diese ursprüngliche Absicht erwies sich aus verschiedenen Gründen als undurchführbar: Neben den empirischen Begriffen mußten deshalb *theoretische Begriffe* zugelassen werden, die nicht vollständig empirisch gedeutet werden konnten. Die Folge war die sog. *Zweistufenkonzeption der Wissenschaftssprache*: die erste Teilsprache bildete die zur Beschreibung der Erfahrungsbasis vollständig interpretierte Beobachtungssprache L_O, die zweite Teilsprache stellte die theoretische Sprache L_T dar, deren Grundbegriffe ungedeutete theoretische Terme bilden. Theoretische Begriffe erhielten eine empirische Deutung allenfalls über Korrespondenz- oder Zuordnungregeln, welche die Verbindung zwischen beiden Teilsprachen herstellen, so daß eine Theorie nur partiell interpretiert war (Stegmüller 1973: 69; Götschl 1980: 639; Carnap 1986, Teil V; eine Zusammenfassung des empiristischen Programms von Carnap findet sich in Stegmüller 1978).

Die Methoden- und Grundlagenlehrbücher der empirischen Sozialforscher berufen sich aber weniger auf den Logischen Empirismus als auf die Methodologie des

[3] Für die entsprechende wissenschaftstheoretische Literatur genügt ein Verweis auf das umfangreiche Werk von Stegmüller.

Kritischen Rationalismus von Popper (1982; für die sozialwissenschaftliche Übernahme vgl. z.B. Schnell/Hill/Esser 1993 oder Prim/Tilmann 1989). Die Ursachen für die breite Rezeption des Kritischen Rationalismus liegen wohl darin, daß die Popper-Schule erstens relativ detailliert ein normatives Regelsystem für die Generierung von Theorien ausgearbeitet hat, daß sich zweitens Popper - im Gegensatz zu Carnap - mit sozialwissenschaftlichen Problemstellungen zumindest ansatzweise beschäftigte[4] und daß drittens die Popper-Schule weitaus weniger Wert legte auf Präzisierung und Formalisierung als Carnap. Nach Popper werden Theorien nicht induktiv aus Daten gewonnen; vielmehr sind Theorien - unabhängig von ihrer nicht begründbaren Entdeckung - als Vermutungen oder Hypothesen zu betrachten, die sich grundsätzlich empirisch bewähren müssen. Theorien bewähren sich dabei in dem Maß, in dem sie Falsifikationsversuchen standhalten. In der Interpretation der Popper-Schule können Theorien niemals als wahr oder wahrscheinlich begründet werden, aber die Bevorzugung gewisser Theorien kann im Lichte ihrer Bewährung gerechtfertigt werden. Der Fortschritt der Wissenschaften besteht darin, Theorien mit möglichst großer Allgemeinheit aufzustellen, und diese ständigen Falsifikationsversuchen auszusetzen, um mit den bewährten Theorien die Welt immer besser beschreiben und erklären zu können (Popper 1982).

Der Kritische Rationalismus unterscheidet sich von den Auffassungen des Logischen Empirismus im wesentlichen in der Ablehnung jeder Induktionslogik und dem Wert von Formalisierungen. *Beide* Schulen betonen aber die Suche nach einem Kriterium für "Wissenschaftlichkeit" und verstehen Wissenschaftstheorie *normativ*. Insbesondere interpretiert die Popper- ebenso wie die Carnap-Schule *Theorien als Aussagenmengen*: "Die Tätigkeit des wissenschaftlichen Forschers besteht darin, Sätze oder Systeme von Sätzen aufzustellen und systematisch zu überprüfen ..." (Popper 1982: 1). "Wissenschaftliche Theorien sind allgemeine Sätze. Sie sind, wie jede Darstellung, Symbole, Zeichensysteme" (Popper 1982: 31).

Das Standardtheorienkonzept hatte mit Beginn der sechziger Jahre zunehmend mit Schwierigkeiten und Problemen zu kämpfen und wurde immer mehr kritisiert und in Frage gestellt (vgl. Kap. 5.1). In den empirischen Wissenschaften ist dieses Konzept aber immer noch dominierend; es wird von den meisten Wissenschaftlern, die der analytischen Wissenschaftsphilosophie nahestehen, weiter vertreten. Wir behalten deshalb zunächst das Aussagenkonzept als metatheoretischen Rahmen bei.

1.2 Modellkonzeptionen

Während der Theoriebegriff - zumindest in der Tradition der empirisch orientierten Sozialwissenschaften - weitgehend mit dem Aussagenkonzept identisch ist, ist der Begriff des Modells "mehrdeutig" (Giesen/Schmid 1977: 82) und wird "ebenso häufig wie vieldeutig verwendet" (Harbordt 1974: 50). Beispielsweise rekonstruiert Mayntz (1967) vier verschiedene Bedeutungen und Verwendungsweisen des Modellbegriffs in der sozialwissenschaftlichen Fachliteratur.

[4] Vgl. z.B. Popper (1987) oder Popper (1972).

In der Literatur zu Computersimulationen wird der Modellbegriff vielfach aus dem alltagssprachlichen Gebrauch entwickelt. Dörner (1984: 337) knüpft an diesen Modellbegriff an, wonach ein Modell "die Replikation eines Realitätsausschnitts, sein Abbild" ist. Ein Modellflugzeug ist ein *Abbild* eines realen Flugzeugs, eine Modelleisenbahn ein *Abbild* eines anderen Realitätsausschnitts. Beim Erstellen von Modellen - der *Modellbildung* - sind also zunächst zwei Objekte beteiligt: das Modell und ein Realitätsausschnitt (auch:"empirisches System"). Abgebildet werden in ein Modell nicht nur *Elemente* des Realitätsausschnitts, sondern auch *Relationen* zwischen den Elementen. Beispielsweise werden in die Modelleisenbahn die Elemente "Lokomotive" und "Schiene" abgebildet und die Relation, daß "Lokomotiven auf Schienen fahren".

Troitzsch (1990: 10) definiert die Beziehung zwischen Wirklichkeitsausschnitt S und Modell M als zweistellige Relation "M ist ein Modell von S". Auch Troitzsch sieht diese *Modellrelation* als *Abbildung* von einem *Urbildbereich* (Realität) in einen *Bildbereich* (Modell) an (Abb. 1.1). Troitzsch weist aber darauf hin, daß die Modellrelation keine Abbildung im streng mathematischen Sinn ist, da sie linkstotal und rechtstotal, aber weder links- noch rechtseindeutig ist. Wir verwenden den Abbildungsbegriff im folgenden deshalb im intuitiven Sinn.

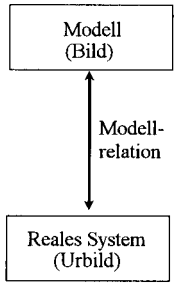


Abb. 1.1: Grundlegendes Modellschema: Reales System, Modell und Modellrelation

Harbordt (1974: 51) beschreibt die vier wesentlichen Merkmale von Modellen wie folgt:

- Die grundlegende Funktion eines Modells ist die *Abbildung* eines Objekts.
- Ein Modell kann als *System* betrachtet werden.[5]
- Das Objekt wird *vereinfacht* abgebildet, indem von seinen "unwichtigen" Merkmalen abstrahiert wird, wobei Wichtigkeit/Unwichtigkeit vom Erkenntnisinteresse abhängt.
- Modellbildung bedeutet immer "*Reduktion von Komplexität*".

Die letzten beiden von Harbordt genannten Modellcharakteristika betonen den neuen Aspekt, nach dem Modellierung ein Abstraktionsprozeß ist, bei dem wichtige Merkmale oder Eigenschaften des modellierten Phänomens identifiziert werden. Mit Hilfe

[5] In der Literatur folgen auf diesen Hinweis meist ausführliche Betrachtungen zur Systemtheorie (z.B. Harbordt 1974, Dörner 1984), auf die wir in diesem Rahmen aber nicht eingehen können.

des abstrakten Modells ist es machbar, daß man sich ausschließlich auf die *relevanten Qualitäten* oder Eigenschaften des Phänomens konzentrieren und die irrelevanten außer acht lassen kann. Was relevant und irrelevant ist, hängt vom Abstraktionszweck und damit vom Erkenntnisinteresse des Modellbauers ab: "Developing a model involves abstracting from reality those components and relationships which are hypothesized as crucial to what is being modeled" (Brody, nach Dawson 1962: 3).

Da bei der Modellierung ein Urbild in ein Bild und damit in *irgendein Medium* abgebildet werden muß, kann man den Bildbereich näher charakterisieren. Nach dem im Modell zur Abbildung verwendeten Bildbereich unterscheidet Troitzsch (1990) vier grundlegende Modelltypen.

Tab. 1.1: Modelltypen und Bildbereiche (nach Troitzsch 1990: 12-13)

Modelltyp	Bildbereich
Realmodell	Existenz des Bildbereichs in der Wirklichkeit (z.B. das "Tiermodell" in der Medizin)
Ikonisches Modell	Konstruktion des Bildbereichs aus der Wirklichkeit (z.B. die Modelleisenbahn)
Verbalmodell	Natürliche Sprache als Bildbereich
Formalmodell	Formale Sprache als Bildbereich (z.B. Differentialgleichungen)

In der Wissenschaft ist mit Modellierung in der Regel die Konstruktion von *formalen Modellen* gemeint. In formalisierten Modellen ist der Bildbereich eine formale Sprache, in der die Repräsentation des abzubildenden Realitätsausschnittes vorgenommen wird. Formale Sprachen sind Kunstsprachen, die gewöhnlich der Mathematik zugerechnet werden, wie z.B. Differentialgleichungen, Graphentheorie, Prädikatenlogik oder Automatentheorie. Formale Sprachen können als *Kalküle* aufgefaßt werden, wenn deren Grundelemente als "bedeutungslose" Zeichen verstanden werden, die mit einer endlichen Menge von Regeln kombiniert und umgeformt werden können. Wenn wir im folgenden von Modellen sprechen, sind immer formale Modelle gemeint.[6] Ziel der formalen wissenschaftlichen Modellierung ist vor allem, mit Modellen Erkenntnisse über die Welt zu bekommen und Untersuchungsergebnisse am Modell auf das reale System zu übertragen: von bestimmten Modelleigenschaften soll auf Eigenschaften des Urbilds zurückgeschlossen werden.

Was bei den eben vorgestellten Modellkonzepten unklar bleibt, ist die Beziehung zwischen Theorie *und* Modell. Für Mayntz (1967) ist der gängigste und zweckmäßigste Modellbegriff der, nach der ein Modell eine *formalisierte* Theorie ist. Im gleichen Aufsatz wird allerdings festgestellt, daß nicht alle Modelle theoretischen Gehalt haben. Mayntz (1967) unterscheidet vielmehr *erklärende* von *deskriptiven* (darstellenden, empirischen) Modellen und ordnet Modelle auf einem Kontinuum an, je nachdem, wieweit in ihnen der Ordnungszweck oder der Erklärungszweck überwiegt. "Je stärker das Modell zu dem deskriptiven oder Ordnungspol tendiert, um so

[6] Ein frühes Beispiel für formale Modellierung in den Sozialwissenschaften ist Herbert Simons (1957) Aufsatzsammlung "Models of Man". Einen moderneren Überblick über die Anwendungsmöglichkeiten formaler Modelle gibt Rapoport (1980).

mehr empirische Erfahrung wird unmittelbar darin enthalten sein. Je mehr es dagegen einem erklärenden Zweck dienen soll, um so mehr theoretische Postulate wird es explizit enthalten. Zwischen den beiden Polen gibt es jedoch graduelle Übergänge" (Mayntz 1967: 15). Ein deskriptives und nicht erklärendes Modell wäre z.B. eine mathematische Funktion, die auf ein bestimmtes Versuchsergebnis paßt und so die Daten lediglich *beschreibt*. Im normalen Gebrauch wird der Modellbegriff aber im Sinn eines erklärenden Modells verwendet, welches gesetzmäßige Beziehungen zwischen den Modellelementen enthält.

Dörner (1984: 343) verwendet implizit ebenfalls "Modell" im Sinn einer *formalisierten* Theorie und stellt die Beziehung zwischen Theorie und Modell als Spiralprozeß dar, bei dem immer neue, an die Realität besser angepaßte (erklärende) Modelle konstruiert werden. Auf diese Weise wird die Theorie immer "vollständiger, differenzierter und zutreffender". Troitzsch (1990) bezweifelt, "ob *die* Theorie bei Dörner in ihrer Entwicklung immer dieselbe bleibt oder ob nicht stattdessen von einer *Folge* immer weniger vager, immer präziserer Theorien die Rede sein sollte" (Troitzsch 1990: 7-8, Hervorhebung Troitzsch).

Die Beziehung zwischen Theorie und Modell bleibt also in der Literatur etwas unklar. Eine eindeutige und präzise Definition von Theorie, Modell und ihrer Relation liefert das unten behandelte strukturalistische Theorienkonzept. In der substanzwissenschaftlichen Literatur wird der Modellbegriff manchmal in einer Bedeutung verwendet, die dem strukturalistischen Modellbegriff entspricht (z.B. Sylvan/Glassner 1985). Danach ist ein Modell für eine Menge von Sätzen eine *mengentheoretische Struktur*, die diese Sätze erfüllt. Diese Modelldefinition geht auf den Logiker Tarski zurück und ist die gängige Modellauffassung in den Formalwissenschaften. Die Relation zwischen Theorie und Modell liegt nun darin, daß alle (empirischen) Strukturen, welche die Sätze einer Theorie erfüllen, Modelle dieser Theorie sind (Sylvan/Glassner 1985: 99-100). Beispielsweise ist das Sonnensystem ein Modell der Newtonschen Gravitationstheorie, da es eine mengentheoretische Struktur aus Elementen (Himmelskörpern) und Relationen darstellt, welche den Gesetzen der Theorie genügt. Hingegen ist ein Vogelschwarm kein Modell der Theorie Newtons, da in dieser Struktur die Gesetzmäßigkeiten nicht erfüllt sind.

Während bei dem Modellkonzept von Dörner und Troitzsch das Urbild ein Realitätsausschnitt ist, ist bei diesem Modellkonzept das *Urbild eine Theorie*, von der viele Modelle existieren. In der Modellrelation "x ist Modell von y" müssen somit bei der Änderung des Modellbegriffs die Argumente vertauscht werden. Im Modellbegriff von Dörner/Troitzsch sind für x Namen von (theoretischen) Modellen einzusetzen und für y Namen von Realitätsausschnitten. Im zweiten Modellbegriff sind hingegen für y Namen von Theorien zu verwenden und für x Namen von empirischen Systemen (oder allgemeiner: mengentheoretischen Strukturen). Um den letzten Modellbegriff verwenden zu können, muß damit eine existierende Theorie vorausgesetzt werden.

Wir haben also zusammenfassend im wesentlichen drei Bedeutungsweisen des Modellbegriffs in der Wissenschaft herausgearbeitet:

1. Hier nicht interessierende deskriptive Modelle ohne erklärenden Gehalt;

2. (Erklärende) Modelle im Sinn einer Abbildung eines Realitätsausschnitts;
3. Modell im Sinn einer (empirischen) Struktur, welche die Sätze einer Theorie erfüllt.

Wir verwenden in diesem Buch die letzten beiden Modellkonzepte und bis zur Einführung der strukturalistischen Modellauffassung den zweiten Begriff.

1.3 Zum Nutzen formaler Modelle

Sowohl in der Wissenschaftstheorie als auch in den Substanzwissenschaften wird immer wieder auf die Vorzüge der Formalisierung und der Verwendung von formalisierten Modellen hingewiesen. In seinen grundlegenden Betrachtungen über die "Ziele und Aufgaben der Wissenschaftstheorie" gibt Stegmüller (1973) seiner Überzeugung Ausdruck, "daß am Ende *jede* Begriffsexplikation in eine mehr oder weniger starke *Formalisierung* einmünden wird" und "daß erst die formalen Kunstsprachen uns die Mittel dafür bereitstellen, *genau zu sagen, was wir eigentlich meinen*...Die Apparatur formaler Kunstsprachen gibt uns erst die Mittel in die Hand, die Probleme *klar zu formulieren* und dadurch überhaupt erst *klar zu sehen* und sie Lösungen zuzuführen, mit denen *ein für den Menschen erreichbares Optimum an Genauigkeit verbunden ist"* (Stegmüller 1973: 14, Hervorhebungen Stegmüller). In den Sozialwissenschaften wird die zu Stegmüller analoge Position beispielsweise von Ziegler vertreten: "Wenn man ernsthaft an der Konstruktion empirisch gehaltvoller soziologischer Theorien interessiert ist, wird man ... das Mittel der Formalisierung einsetzen müssen, und zwar allein schon deshalb, weil andere Vorgehensweisen wenig Erfolg versprechen" (Ziegler 1972: 291). Für eine Formalisierung soziologischer Theorien tritt ebenfalls Hummell ein: "in vielen Fällen besteht in der Konstruktion formalisierter Aussagensysteme die einzige Chance, theoretische Erörterungen überhaupt durchführen zu können" (Hummell 1972b: 136).

In seinem Aufsatz "Warum Formalisierung in der Wissenschaft erwünscht ist" plädiert der Wissenschaftsphilosoph Patrick Suppes (1983: 27ff.) mit folgenden Argumenten für eine (mengentheoretische) Formalisierung und Axiomatisierung[7] von Begriffen und Theorien:

1. *Explizitheit*: Mit der Formalisierung werden die verwendeten Begriffe in *expliziter* Weise herausgearbeitet. Vor der expliziten Analyse des Wahrscheinlichkeitsbegriffs gab es beispielsweise über die elementarsten Eigenschaften der Wahrscheinlichkeit Verwirrung, die erst mit der Kolmogoroffschen Formalisierung beseitigt wurden.
2. *Standardisierung*: Die explizite Formalisierung einer Theorie bedingt die "Standardisierung der Terminologie und der Methoden der begrifflichen Analyse, die dadurch in verschiedenen Zweigen der Wissenschaft erreicht wird". Dadurch wird die Kommunikation zwischen wissenschaftlichen Disziplinen erleichtert und die Einheit der Wissenschaften gefördert.

[7] Die mengentheoretische Axiomatisierung von Theorien ist genau genommen *ein ganz bestimmter* Formalisierungsansatz, der dem späteren strukturalistischen Vorgehen entspricht. Die Argumente von Suppes sind aber relativ allgemein gehalten, so daß sie weitgehend für jede Art der Formalisierung zutreffen.

3. *Allgemeinheit*: Formalisierung eliminiert unwesentliche Züge in einer Theorie und erlaubt es, "den Wald trotz lauter Bäumen zu sehen".
4. *Objektivität*: Formalisierte Theorien liefern einen weitaus höheren Grad an Objektivität als nicht-formalisierte Theorien. Der Wert solcher Formalisierung kann wesentlich sein in Wissensgebieten, wo schon über die elementarsten Begriffe große Kontroversen bestehen.
5. *Abgeschlossenheit der Annahmen*: "Formalisierung stellt einen Weg dar, im Wald der impliziten Annahmen und dem umgebenden Dickicht der Verwirrung den festen Grund auszumachen, der für die betrachtete Theorie benötigt wird".
6. *Minimale Annahmen*: Formalisierung ermöglicht eine objektive Analyse der minimalen Annahmen, welche zur Formulierung der Theorie nötig sind. "Ich glaube, daß man aus einem direkten ästhetischen Bedürfnis heraus eine Menge wechselseitig unabhängiger Annahmen, die abgeschlossen sind, für die Formulierung einer Theorie finden möchte".

Substanzwissenschaftler argumentieren ähnlich, betonen aber eher den allgemeinen Präzisierungseffekt und die Möglichkeit, intuitive und schwer überschaubare Schlußfolgerungen durch präzise Deduktionen zu ersetzen. Je nach Autor werden bestimmte Aspekte hervorgehoben; wir fassen die wichtigsten knapp zusammen:

1. Formalisierung erlaubt es, *Ableitungen vorzunehmen*, vor denen die Mittel der natürlichen Sprache versagen, insbesondere bei komplexen Aussagesystemen (Troitzsch 1990: 32). Ziegler (1972: 14-15) sieht das zentrale Argument für eine Formalisierung darin, "daß die in unserer wissenschaftlichen Alltagssprache vorhandenen 'Ableitungsregeln' bei vielen, vermutlich sogar den meisten Aussagen, die in den Sozialwissenschaften verwendet werden, nicht ausreichen, um überhaupt präzise deduzieren zu können". Die Ableitungen werden erleichtert und besser kontrollierbar, da man ohne inhaltliche Überlegungen rein mechanisch nach den Regeln der verwendeten formalen Sprache (des Kalküls) neue Sätze herleiten kann (Opp 1976: 320-321). Bunge (1983: 64) spricht von der "deduktiven Kraft, die einer verbalen Doktrin fremd ist".
2. Formalisierung führt zu einer *Präzisierung* der verwendeten Theorie in dem Sinn, daß zum einen begriffliche Vagheiten und Mehrdeutigkeiten leichter entdeckt werden und zum anderen begriffliche Zusammenhänge eindeutig spezifiziert werden müssen (Opp 1976: 322, Harbordt 1974: 243). Durch die Übersetzung in eine formale Sprache werden die ursprünglich vagen Aussagen und Zusammenhänge einer verbalen Theorie modifiziert und präzisiert.
3. Formalisierung trägt zur *Systematisierung* und *Klärung der logischen Struktur* einer Theorie bei. Es lassen sich klare Beziehungen der logischen Folgerung oder Implikation zwischen Aussagen herstellen (Hummell 1972a: 32). Inbesondere bei axiomatischen Darstellungen kann streng zwischen Axiomen und Theoremen unterschieden werden (Opp 1976: 323). "Die Axiomenmenge ist dabei so beschaffen, daß sämtliche Theoreme logische Konsequenzen der Axiome sind. Für axiomatisch-deduktive Systeme wird die Relevanz der logischen Beziehung der Ableitbarkeit in besonders eindrücklicher Weise deutlich" (Hummell 1972a: 32).

In der sozialwissenschaftlichen Literatur finden sich oft Argumente, die *gegen* eine Formalisierung sprechen, wie z.B. die folgende Bemerkung: "Was bei der Formalisierung notwendig verloren geht, ist die Aura des assoziativ Mitgedachten, die verbale Aussagen stets umgibt, über ihren ausdrücklichen Gehalt hinaus mit Wirklichkeit sättigt und auch in Grenzfällen noch zutreffend erscheinen läßt" (Mayntz 1967: 28, vgl. auch Harbordt 1974: 271). Solche Kritikpunkte können wir nicht nachvollziehen. Da eine unerläßliche Voraussetzung wissenschaftlichen Arbeitens das "Bemühen um sprachliche Klarheit" ist, schließen sich Vertreter solcher Positionen aus dem Bereich wissenschaftlicher Tätigkeit aus (vgl. Stegmüller 1973: 5). Wenn man daran interessiert ist, "die Aura des assoziativ Mitgedachten" zu erhalten, ist schöngeistige Literatur ein geeigneteres Betätigungsfeld als Wissenschaft.

2. Grundzüge der Computermodellierung

Computermodelle sind eine echte Teilmenge formaler Modelle, und Computermodellierung ist die Konstruktion und Ausführung von Modellen auf dem Computer. Computermodelle haben zusätzlich zu den Charakteristika formaler mathematischer Modelle die Eigenschaft, daß sie *von einem Rechner interpretiert und ausgeführt* werden können.[8] Das Repräsentationsmedium des Bildbereichs - die formale Sprache - bilden in diesem Fall Programmiersprachen oder eigens entwickelte Modellbeschreibungssprachen (Möhring 1990: 11). Abb. 2.1 zeigt eine Differentialgleichung als formales Modell und die programmiersprachliche Repräsentation als Computermodell.

Formales Modell (Differentialgleichung):

$$\ddot{x} = -2\zeta\omega\dot{x} - \omega^2 x + \omega^2 f$$

mit x(0) = ẋ(0) = 0

Computermodell (FORTRAN-ähnliche Sprache):

```
X2D = -2.0*ZETA*OMEGA*X1D-X*OMEGA**2+OMEGA**2*F
X1D = INTGRL(0.0, X2D)
X   = INTGRL(0.0, X1D)
```

Abb. 2.1: Formales Modell und Computermodell (Kheir, nach Möhring 1990: 12)

Mit *Computersimulation* ist meist ein dynamischer Aspekt gemeint. Sie betrifft Modelle, in denen die zeitliche Entwicklung eines Systems repräsentiert ist. Troitzsch (1990: 178) faßt den Simulationsbegriff weiter, nämlich als "Experimentieren mit Modellen", Kreutzer (1986: 5) als "experimental technique for exploring 'possible worlds' through computational processes" und für Möhring (1990: 13) ist Simulation konkret "Beobachtung von Modellverhalten über einen längeren Zeitraum".[9]

In Erweiterung des Schemas von Abb.1.1 lassen sich die Relationen zwischen empirischem System, Modell und Computerprogramm grafisch in der folgenden Abbildung verdeutlichen.

---

[8] Computermodelle haben im Gegensatz zu mathematischen Modellen aber auch die Eigenschaft, daß sie Teile enthalten, die für das eigentliche Modell völlig irrelevant sind (vgl. Kap. 3.1.2).

[9] Der Simulationsbegriff von Troitzsch, Kreutzer und Möhring ist insofern "weiter", als er Simulation nicht einengt auf dynamische Modelle. Zwar haben die meisten Computermodelle dynamischen Charakter, aber es gibt auch nicht-dynamische, statische Modelle (vgl. Tab. 2.1, S.35).

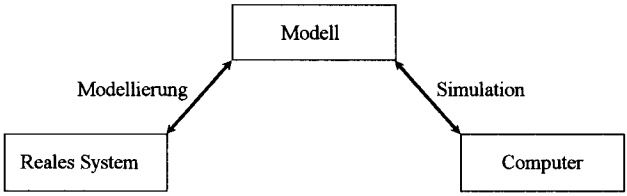


Abb. 2.2: Modellbildung und Simulation (nach Zeigler 1976: 4)

Dieses Schema ist allerdings insofern kritisch zu betrachten, als es zum einen möglich ist, daß es zu einem Computerprogramm kein verbales oder mathematisches Gegenstück gibt, so daß das Programm unmittelbar das Modell ist. Zum andern läßt das Schema die Relation von Modell und Theorie außer acht und es ist denkbar, daß eine Theorie direkt in einem Programm ausgedrückt wird. Auf diese Aspekte wird weiter unten eingegangen.[10]

2.1 Elementare Eigenschaften von Computermodellen[11]

Ein grundlegendes Merkmal von Computermodellen - oder allgemeiner: Computerprogrammen - ist, daß sie Eingabedaten entgegennehmen, diese verarbeiten und Ausgabedaten produzieren. Das Programm verändert die eingegebenen Daten gemäß bestimmter Regeln und gibt Daten an der Computerperipherie (Bildschirm, Drucker etc.) aus. Die Programmregeln sind so flexibel, daß sie auf eine ganze Menge von Eingaben angewandt werden können. Wenn wir vorerst das Computermodell M als Black-Box betrachten, dann nimmt M also eine Menge von *Eingabedaten (Input)* entgegen und liefert eine Menge von *Ausgabedaten (Output)*. Wir bezeichnen die Menge der möglichen Modellinputs mit I_M und die Menge der möglichen Outputs mit O_M.[12] Die $i_M \in I_M$ brauchen vorerst nicht näher spezifiziert werden: es können Zahlen (z.B. Vektoren, Matrizen) oder beliebige andere Zeichenketten sein - im Extremfall auch umgangssprachliche Sätze (vgl. Colby 1973, 1975 und das Beispiel S.22). Die Daten können ferner von realen empirischen Systemen stammen, es kann sich aber auch um künstliche oder hypothetische Daten handeln, wie dies besonders bei sozialwissenschaftlichen Simulationen oft der Fall ist.

I_M kann leer sein, nämlich dann, wenn M keine Eingabewerte verlangt - was allerdings nicht der Regelfall ist. Analog ist es denkbar, aber wenig sinnvoll, daß O_M leer ist, nämlich dann, wenn M keine Outputdaten erzeugt.

Die Anwendung des Modells M auf i_M liefert o_M, so daß wir folgendes *Input-Output-Schema* erhalten.

---

[10] Andere Schemata, auf die wir hier nicht weiter eingehen, diskutiert Troitzsch (1990).
[11] Vgl. zu diesem Abschnitt insbesondere Zeigler (1976) und Kobsa (1984).
[12] Wir lassen hier einfachheitshalber den Zeitaspekt außer acht.

Abb. 2.3: Input-Output-Schema für ein Modell M

Wir nennen das Paar $\langle i_M, o_M \rangle$ ein *Input-Output-Paar* oder *I/O-Paar*. Die Wiederholung dieses Vorgangs mit verschiedenen i_M ergibt eine Menge solcher I/O-Paare, die wir als *I/O-Relation* R_M bezeichnen: $R_M \subseteq I_M \times O_M$.

R_M ist also die Menge aller I/O-Paare. Ist die I/O-Relation rechtseindeutig, d.h. bildet sie einen beliebigen Eingabewert auf genau einen Ausgabewert ab, so sprechen wir von einer *I/O-Funktion*. Dies ist der Fall bei deterministischen Modellen, in denen keine Zufallsereignisse vorkommen. In stochastischen Modellen ist es möglich, daß es für ein bestimmtes i_M unterschiedliche o_M gibt. Wir beschränken uns im folgenden auf deterministische Modelle, die I/O-Funktionen liefern.

In diesem Fall kann das Modell als Funktion f_M von der Menge der Inputdaten in die Menge der Outputdaten betrachtet werden (Claus 1986: 378-379):

f_M: $I_M \rightarrow O_M$

Mit *Modell-* oder *I/O-Verhalten* ist die Umwandlung des Modellinputs in den Modelloutput gemeint, d.h. die Generierung von Outputdaten aus Inputdaten. Das vom Modell abgebildete empirische System S kann nun ebenfalls als Black-Box betrachtet werden, das aus Inputwerten $i_S \in I_S$ Outputwerte $o_S \in O_S$ generiert. Ziel des Modells ist zunächst, diesen Realitätsausschnitt S nachzubilden, so daß das Modellverhalten dem Verhalten des empirischen Systems gleich oder zumindest ähnlich ist. Wir vergleichen folglich das Verhalten des Modells mit dem Verhalten des empirischen Systems, was in Abb. 2.4 veranschaulicht wird (nach Harbordt 1974: 161).

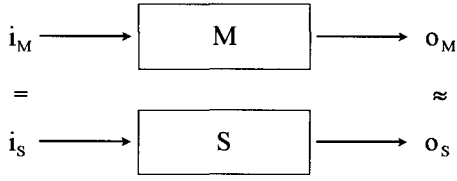


Abb. 2.4: Vergleich des I/O-Verhaltens von Modell M und empirischem System S. "≈" repräsentiert die Ähnlichkeitsrelation.

Bei der Modellkonstruktion strebt man in der Regel danach, daß die Beobachtung eines bestimmten I/O-Paares in S einem gleichen oder zumindest ähnlichen I/O-Paar in M korrespondieren soll. Umgekehrt sollte jedem I/O-Paar in M ein gleiches oder ähnliches Paar in S korrespondieren. Konkreter formuliert lautet letzteres: wir geben in das Modell M genau die Inputdaten, die für S zu einem bestimmten Zeitpunkt gültig sind (z.B. in ein Bevölkerungsmodell demographische Daten von 1960) und sollten von M gleiche oder zumindest ähnliche Werte zu S für einen späteren Zeitpunkt als Outputdaten erhalten (z.B. die gleiche oder eine ähnliche Geschlechts-/Altersverteilung für 1994). Formal ausgedrückt sollte M idealerweise für alle I/O-Relationen

sowohl vollständig als auch korrekt sein. M ist *vollständig*, wenn jede I/O-Relation des empirischen Systems die korrespondierende I/O-Relation in M zur Folge hat. Jede beliebige empirische I/O-Relation zwischen i_S und o_S müßte bei Vollständigkeit des Modells also die analoge I/O-Relation in M liefern, genauer (mit "≈" als Symbol für die Ähnlichkeitsrelation):
für alle $i_S \in I_S$, $o_S \in O_S$: wenn i_S zu o_S führt, dann gibt es $i_M \in I_M$, $o_M \in O_M$, so daß $i_M = i_S$ und $o_M \approx o_S$.

Umgekehrt heißt M *korrekt*, wenn jede I/O-Relation des Modells, bei dem der Modellinput mit dem Input des empirischen Systems identisch ist, bei Eintreten des Systemoutputs einen ähnlichen Modelloutput liefert (vgl. hierzu auch Kobsa 1984: 53). Es müßte also genauer gelten:
für alle $i_M \in I_M$, $o_M \in O_M$: wenn i_M zu o_M führt und es gibt $i_S \in I_S$, so daß $i_M = i_S$, dann $o_M \approx o_S$.

Vollständigkeit und Korrektheit für *alle* I/O-Relationen sind für die Modellierung zu starke Forderungen. Sie würden eine vollständige Verhaltenskopie des empirischen Systems erzeugen. Die oben gelieferten Definitionen sind deshalb auf den modellierten Ausschnitt des empirischen Systems und die dort betrachteten Objekte und Relationen einzuschränken, so daß die *Vollständigkeits- und Korrektheitsforderung* nur *partiell* für diese Teilmenge gilt.

Vollständigkeit und Korrektheit beziehen sich auf die Adäquatheit, also Gültigkeit oder *Validität* von Modellen. Validität wird i.a. als *notwendig* angesehen, um von Modelleigenschaften auf Eigenschaften des empirischen Systems rückzuschließen. Validität im Sinne von *vollständiger Identität* der I/O-Paare von Modell und empirischem System ist in der Regel eine zu starke Forderung, die Outputs sollten aber zumindest *ähnlich* sein. Das Problem ist dann die Spezifizierung der Ähnlichkeitsrelation und es wird noch verschärft, wenn das Modell Wahrscheinlichkeitsfunktionen verwendet, die zu Zufallsschwankungen im Modellverhalten führen.

Zur Überprüfung der Validität bieten sich für numerische Modelle - d.h. für Modelle, deren Ein- und Ausgaben Zahlen darstellen - statistische Tests an: man läßt z.B. das Modell und das reale System Stichproben von Verhaltensprotokollen erzeugen[13] und vergleicht die entsprechenden I/O-Paare des Modells und des realen Systems mit den üblichen statistischen Verfahren (für einen Überblick über diese Verfahren vgl. Harbordt 1974). Eine andere Vergleichsvariante sind bereichsspezifische Turing-Tests, die vor allem bei nicht-numerischen Modellen verbreitet sind. Vereinfacht gesagt besteht ein Modell M den Turing-Test genau dann, "wenn ein kompetenter Beobachter nicht in der Lage ist, eine Verhaltenssequenz des Urbildes von der Verhaltenssequenz des Modells zu unterscheiden, wenn er also Verhaltenssequenzen aus diesen beiden verschiedenen Quellen verwechselt" (Dörner 1984: 347).[14]

---

[13] Da das reale System oft nicht manipulierbar ist, muß man meist von diesem ausgehen und die empirisch aufgetretenen Eingabewerte dann zum Modellinput machen.

[14] Der Mathematiker Alan Turing überlegte sich die allgemeine Fassung dieses Tests 1950 im Zusammenhang mit der Frage, ob Maschinen denken können. Grob gesprochen, ist auf diese Fragestellung nach Turing positiv zu reagieren, wenn das Antwortverhalten der Maschine in einem Frage-Antwort-Spiel von dem eines Menschen nicht zu unterscheiden ist. Für nähere Details vgl. Boden (1987) oder Abelson (1968).

Der Turing-Test eignet sich aufgrund mehrerer Eigenschaften nur bedingt für Modelltests (z.B. wer bestimmt die Kompetenz des Beobachters, welches Kriterium zieht er für sein Unterscheidbarkeits/Nicht-Unterscheidbarkeits-Urteil heran?). Abelson (1968: 318-320) stellt einen erweiterten Turing-Test vor, und in der Literatur finden sich weitere Varianten dieses Tests. Mehrere dieser Modifikationen wendet z.B. Colby (1975) in einem bekannten psychologischen Modell an, welches paranoides Verhalten simuliert. Das Modell nimmt als Inputs umgangssprachliche Sätze entgegen und produziert solche als Outputs, z.B. (Colby 1975: 78):

Input Interviewer: HAVE YOU BEEN ACTIVELY TRYING TO AVOID THE UNDERWORLD?
Output Modell: NO ONE HAS ANY POWER OVER GANGSTERS.

Mehrere modifizierte Turing-Tests wurden von dem Modell "bestanden". Bei einem dieser Tests sollten z.B. ausgewählte und angesehene Psychiater Interviewtranskripte, die von dem Modell stammten und solche, die von einem wirklichen paranoid Kranken herrührten, identifizieren. Die Gutachter nahmen zu 51% die richtige und zu 49% die falsche Identifikation vor, was statistisch interpretiert auf dem 95%-Konfidenzintervall einem Zufallsergebnis entspricht. Mit anderen Worten: die Experten konnten nicht unterscheiden, welche Antworten von dem Menschen und welche von der Maschine kamen.

Fragen der Gültigkeit von Modellen stehen nicht im Mittelpunkt dieser Arbeit und sollen hier nicht weiter behandelt werden. Die Gültigkeit spielt zudem bei vielen Computermodellen keine entscheidend große Rolle, da diese oft die Funktion von Ideenfindern und "formalisierten Gedankenexperimenten" haben (Ziegler 1972). "Das Gedankenexperiment dient dazu, die potentielle Erklärungskraft von Theorien zu überprüfen, d.h. festzustellen, ob sich bestimmte Aussagen tatsächlich aus gegebenen Prämissen ableiten lassen. Dagegen kommt ihm keine empirische Beweiskraft zu. Man wird jedoch Zeit und Kosten für theoretisch fruchtlose Untersuchungen sparen, wenn man von vornherein logisch nicht schlüssige 'Erklärungen' aussondert und nicht versucht, sie empirisch 'zu überprüfen'" (Ziegler 1972: 89). Computersimulationen können deshalb als moderne Form des Gedankenexperiments betrachtet werden, in der Theorien oder Hypothesen konkret und unter kontrollierten Bedingungen "durchgerechnet" werden können, ohne sich zunächst um empirische Fragestellungen oder Validität zu kümmern.

Wir haben das Modell bislang als Black-Box betrachtet im Sinne einer Funktion, die Inputwerte nimmt und Outputwerte liefert:
$f_M(i_M)=o_M$.
Wird das Modell nur nach seinem I/O-Verhalten beurteilt und bildet es lediglich die I/O-Relationen des empirischen Systems ab, so ist das Modell bestenfalls ein *Verhaltensmodell*. Ein Verhaltensmodell ist seinem Urbild allenfalls gleich in der Art und Weise, wie es auf Inputs reagiert. Das gleiche I/O-Verhalten kann aber von einem ganz anderen Modell ebenfalls erzeugt werden, unter Umständen gibt es unendlich viele Modelle $M_1,...,M_n,...$ die sich völlig gleichartig verhalten. Man nennt Modelle, die das gleiche I/O-Verhalten produzieren, *funktional äquivalent* (Abb. 2.5). Allge-

mein heißen zwei Prozesse funktional äquivalent, wenn sie im mathematischen Sinn dieselbe Funktion realisieren (Görz 1988: 73).[15]

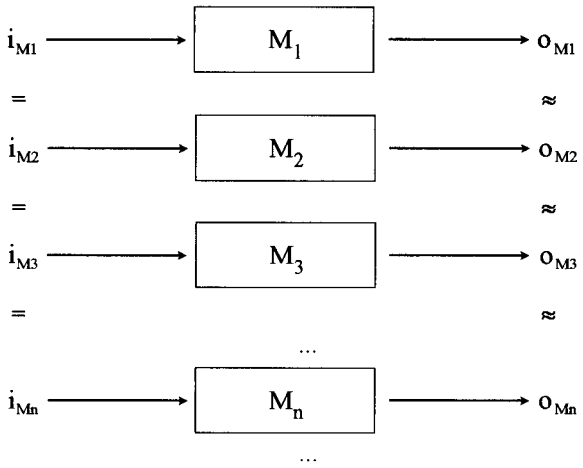


Abb. 2.5: Funktional äquivalente Modelle $M_1,...,M_n...$

Während ein Verhaltensmodell also seinem Urbild nur gleich ist oder gleich sein sollte im Hinblick auf die Art und Weise, wie es auf äußere Reize reagiert, ist ein *Strukturmodell* seinem Urbild zusätzlich auch gleich hinsichtlich der betrachteten Elemente und Relationen, d.h. es bildet die betrachtete Form und das Gefüge des Urbilds ab (für eine genaue Definition des Strukturbegriffs vgl. Kap. 5.2.1, S.104). Ein Strukturmodell ist damit immer auch ein Verhaltensmodell, aber nicht umgekehrt (Dörner 1984: 342).

Unter Verwendung der aus der mathematischen Modelltheorie stammenden Begriffe Homomorphie und Isomorphie läßt sich die Modellrelation für Strukturmodelle spezifizieren. Harbordt (1974: 58) führt die beiden Begriffe informell wie folgt ein:

Ein Modell M ist zu einem empirischen System S *isomorph* genau dann, wenn (gdw) gilt:

(1) Jedem Modellelement $m \in M$ ist ein $s \in S$ eindeutig zugeordnet und umgekehrt;
(2) Jeder Relation in S ist eine Relation in M eindeutig zugeordnet und umgekehrt;
(3) Einander zugeordnete Relationen enthalten nur einander zugeordnete Elemente.

Ein Modell M ist zu einem empirischen System S *homomorph* gdw gilt:

(1) Jedem Modellelement $m \in M$ ist ein $s \in S$ eindeutig zugeordnet, aber nicht umgekehrt;
(2) Jeder Relation in M ist eine Relation in S eindeutig zugeordnet, aber nicht umgekehrt;

[15] Görz (1988: 73) verweist auf ein anschauliches Beispiel von Colby: Alle Mausefallen sind funktional äquivalent. Es gibt zwar verschiedene Mechanismen, um Mäuse zu fangen, doch die Bezeichnung "Mausefalle" besagt, was ihnen gemeinsam ist: jede hat als Input eine lebendige Maus und als Output eine tote.

(3) Einander nach (2) zugeordnete Relationen enthalten nur einander nach (1) zugeordnete Elemente.

Die Isomorphie-Relation gilt in *beide* Richtungen, Homomorphie nur vom Modell zum Urbild. In einer isomorphen Modellrelation würden sich also alle Elemente und Relationen des Urbilds im Modell vollständig wiederfinden und umgekehrt. In einer homomorphen Modellrelation hingegen sind nicht alle Elemente und Relationen des Urbilds im Modell repräsentiert. Harbordt (1974: 59) verweist darauf, daß analog wie uneingeschränkte Vollständigkeit und Korrektheit auch Isomorphie eine ungeeignete Zielvorstellung ist, da das Modell die gleiche Komplexität annehmen würde wie das empirische System. Dies widerspricht dem Vereinfachungscharakter von Modellen. Man begnügt sich deshalb in der Regel mit strukturähnlichen, homomorphen Modellen (Harbordt 1974: 59).

Faßt man die Modellrelation als Homomorphismus auf, kann man andererseits aber nicht mehr von Merkmalen des Urbilds auf Modellmerkmale schließen, womit die Tatsache verlorengeht, daß empirische Systeme in Modelle abgebildet werden. Dörner (1984) bezeichnet die Modellrelation in Strukturmodellen deshalb besser als "*partiellen Isomorphismus*", bei dem die Isomorphierelation nur "bezüglich bestimmter, ausgewählter Merkmale" gilt. Bezüglich dieser vom Modellierer ausgewählten Merkmale herrscht dann "eine umkehrbar eindeutige Abbildung, bei der alle Relationen erhalten bleiben" (Tack, nach Dörner 1984: 337).

In einer so definierten Modellrelation steht im übrigen *jedes* der Systeme Urbild und Bild in Modellrelation zum anderen. Die Modellrelation ist also symmetrisch, so daß nicht unbedingt festgelegt werden muß, welches als Modell für das andere dient. Die Symmetrie der Modellrelation macht es überflüssig, Bild und Urbild zu identifizieren: das in Kap. 1.2 behandelte Problem, welches System wir als Modell bestimmen, ist somit eine reine Frage der Pragmatik.

Die Teilmenge des Urbildbereichs, deren Elemente keine Bilder im Modell haben, bezeichnet man als *Präteritionsklasse* des Modells. Bei der Modelleisenbahn wäre dies z.B. der Dampfkessel einer Lokomotive. Die Teilmenge des Modells, deren Elemente nicht Bilder von Elementen des empirischen Systems sind, bezeichnet man als *Abundanzklasse* des Modells. Der Schleifbügel des Stromabnehmers bei der Modelleisenbahn hat z.B. kein Äquivalent im empirischen System (Troitzsch 1990: 14; Dörner 1984: 338). Abb. 2.6 veranschaulicht diese Beziehung grafisch.

Bei der Frage nach der Gültigkeit von Strukturmodellen kommt zu der Prüfung des I/O-Verhaltens noch die Prüfung auf Strukturgleichheit oder zumindest -ähnlichkeit von Modell und Urbild hinzu. Auf diese zum Teil recht diffizilen und kontrovers diskutierten Möglichkeiten kann hier nicht eingegangen werden. Einige davon werden besprochen in Dörner (1984) oder Harbordt (1974).

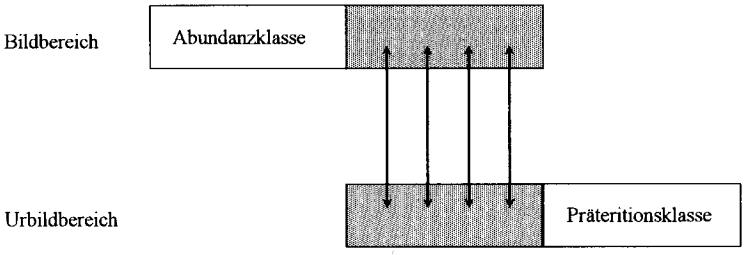


Abb. 2.6: Partieller Isomorphismus beim Strukturmodell mit Abundanz- und Präteritionsklasse (nach Troitzsch 1990: 14). Im abgedunkelten Bereich liegen die einem partiellen Isomorphismus genügenden Elemente und Relationen.

Damit sind die für unsere Zwecke wichtigsten Grundmerkmale der Computermodellierung angesprochen. Ausführlicher wird dieses Thema behandelt in Harbordt (1974), Kreutzer (1986) und Zeigler (1976). Zeigler (1976) entwickelt eine formale Modellierungs- und Simulationstheorie.

2.2 Effektive Verfahren und Algorithmen[16]

Das I/O-Verhalten von Computermodellen wird durch den im Computer realisierten Algorithmus bestimmt. Ein *Algorithmus* oder *effektives Verfahren* ist eine mechanische Verarbeitungsvorschrift, die angibt, wie Eingabedaten schrittweise in Ausgabedaten umgewandelt werden.

Algorithmen besitzen drei Eigenschaften (Kleinknecht/Wüst 1976: 57-58):

1. *Determiniertheit*, d.h. die Verarbeitungsvorschrift ist bis in alle Einzelheiten eindeutig festgelegt, und Wiederholung mit gleichen Eingabewerten und Startbedingungen liefert das gleiche Ergebnis;
2. *Allgemeinheit*, d.h. ein Algorithmus dient zur Beantwortung einer ganzen Klasse von Fragestellungen und nicht nur eines speziellen Problems;
3. *Endlichkeit*, d.h. ein Algorithmus ist durch einen endlichen Text beschreibbar und bricht nach endlich vielen Schritten mit einem Ergebnis ab.

Die Idee, ein mechanisches Verfahren oder einen Algorithmus zur Durchführung einer Aufgabe zu haben, existiert schon seit tausenden von Jahren. Eine exakte Bestimmung des Algorithmusbegriffs gelang aber erst Anfang dieses Jahrhunderts im Rahmen mathematischer Grundlagenforschung. Anlaß für diese Explikation war das um 1900 aufgeworfene Hilbertsche Entscheidungsproblem.[17] Die Frage von Hilbert bestand darin, ob es einen Algorithmus geben kann, der für eine beliebige Aussage in einem gegebenen formalen System entscheidet, ob die Aussage richtig oder falsch ist. Gäbe

[16] Vgl. zu diesem Kapitel insbesondere das ausführliche Kap. 3 "Theorie der Algorithmen" in Goldschlager/Lister (1990).

[17] Das Hilbertsche Entscheidungsproblem war Teil eines Programms der Neubegründung der Mathematik in Form kalkülisierter Axiomensysteme. Die axiomatische Fundierung der Mathematik wurde infolge der Entdeckung verschiedener Antinomien als notwendig erachtet. Einen guten Überblick über diese Thematik gibt Krämer (1988).

es einen solchen Algorithmus, dann ließe sich jedes wohldefinierte Problem einfach durch eine Algorithmusausführung lösen. Das Beweisproblem bestand zunächst in der "Operationalisierung" des Algorithmusbegriffs. Einer der Versuche, "Algorithmus" präzise festzulegen, geht auf Alan M. Turing zurück. Turings Idee war es, den intuitiven Algorithmusbegriff mit Hilfe einer Menge von Anweisungen an eine einfache Maschine zu erfassen. Zur Lösung des Entscheidungsproblems konstruierte Turing (1937) ein gedankliches Maschinenmodell - das später als Turing-Maschine bezeichnet wurde - und stellte die folgende Behauptung auf.

TURING-THESE

Jeder Algorithmus im intuitiven Sinn kann auf einer Turing-Maschine ausgeführt werden.[18]

Die Turing-These ist kein Theorem, sondern der Vorschlag einer *Begriffsexplikation.* Diese Begriffsexplikation von "algorithmisch berechenbar" durch "mit Turing-Maschinen berechenbar" konnte bislang nicht widerlegt werden in dem Sinn, daß es etwas gibt, was intuitiv algorithmisch ist und nicht auf einer Turing-Maschine ausführbar ist. Alle im Verlauf der Geschichte erdachten Algorithmen lassen sich durch die Turing-Maschine erfassen. Mathematiker sind von der Gültigkeit der Turing-These so sehr überzeugt, daß mathematische Beweise, die sich auf sie beziehen, akzeptiert werden (Claus 1986: 97).

In dem so festgelegten Sinn - daß es für jeden Algorithmus im intuitiven Sinn eine Turing-Maschine gibt - konnte Turing zeigen, daß kein Algorithmus von der Art existiert, nach dem Hilbert strebte. Church, Kleene, Post und Gödel kamen in den dreißiger Jahren zu dem gleichen Ergebnis, wobei jeder "Algorithmus" in unterschiedlicher Weise definierte. Diese unterschiedlichen Definitionen werden als gleichwertig aufgefaßt (Church-Turing-These), und bis heute gibt es kein einleuchtendes Gegenbeispiel dafür, daß alle vernünftigen Algorithmusdefinitionen nicht gleichwertig und gleichbedeutend seien.

Turing-Maschinen sind theoretisch bedeutsam, haben praktisch aber keine Relevanz, da heutige Computer in der Regel als von-Neumann-Maschinen gebaut werden. Das charakteristische Prinzip des von-Neumann-Rechners ist, daß zur Lösung eines Problems von außen ein Programm eingegeben werden muß, das im selben Speicher liegt wie die Daten. Berechnungen werden schrittweise, eine nach der anderen, durchgeführt. Den Zusammenhang zwischen Turing- und von-Neumann-Maschinen kann man sich in etwa wie folgt vorstellen. Ein von-Neumann-Computer, der mit einem festen Programm arbeitet, ist durch eine Turing-Maschine beschreibbar. Umgekehrt gibt es zu jeder Turing-Maschine ein Computerprogramm, das diese simuliert. Turing-Maschinen und Computer mit potentiell unendlichem Speicher sind also bezüglich der durch sie erzeugten Funktionenklasse äquivalent (Claus 1986: 504). Dies bedeutet, daß ein Computer alle Turing-berechenbaren Probleme behandeln kann, so daß *ein Computer praktisch alle realen Prozesse ausführen kann, die intuitiv als effektive Verfahren oder algorithmisch bezeichnet werden können.* Umge-

[18] Die Turing-Maschine kann hier nicht dargestellt werden. Eine genaue Beschreibung findet sich z.B. in Reischuk (1991) oder in Krämer (1988). Eine populärere Darstellung mit Hintergründen und Auswirkungen dieses abstrakten Modells gibt Hopcroft (1984).

kehrt ist alles, was auf dem Computer realisierbar ist, algorithmisch. Darüber hinaus kann jeder Algorithmus, der auf einem bestimmten Computer realisiert wird, auf jedem anderen Computer realisiert werden: jeder Computer ist allen anderen in dem Sinn gleichgestellt, daß alle im Prinzip die gleichen Aufgaben ausführen können (Universalität).

Die Bearbeitungsmöglichkeit algorithmisch erschließbarer Probleme durch Computer hat Konsequenzen für das Modellierungspotential des Computers. "Da der Mensch, die Natur und selbst die Gesellschaft mit Verfahren arbeiten, die zweifellos in der einen oder anderen Hinsicht 'effektiv' sind, folgt daraus, daß ein Computer zumindest den Menschen, die Natur und die Gesellschaft in allen verfahrensmäßigen Aspekten imitieren kann. Wir müßten also immer imstande sein, wenn wir ein Phänomen insofern zu verstehen glauben, als wir dessen Verhaltensregeln kennen, unser Verständnis in Form eines Computerprogramms auszudrücken" (Weizenbaum 1978: 95). Ein guter Test, ob ein Phänomen - sei es physischer, psychischer oder sozialer Natur - verstanden wird, ist deshalb der Versuch, für dieses einen Algorithmus zu finden und in einem Programm nachzubauen.

Die Turing-These impliziert nicht, daß sich *alles* in Begriffen eines effektiven Verfahrens beschreiben läßt. Dies trifft sogar auf die *wenigsten* Problemstellungen zu.

- Bestimmte Probleme sind dadurch charakterisiert, daß wir nicht nur *keinen Algorithmus kennen*, sondern es läßt sich sogar beweisen, daß es *keinen Algorithmus geben kann.* Man spricht im diesem Fall auch von Nicht-Berechenbarkeit, im positiven Fall - also wenn es einen Algorithmus gibt - von Berechenbarkeit. Ein berühmtes Beispiel, das für die Arithmetik zugleich eine (negative) Antwort auf das Hilbertsche Entscheidungsproblem liefert, ist das Unvollständigkeitstheorem von Gödel 1931. Es besagt, daß es in einem beliebigen widerpruchsfreien arithmetischen System immer wahre arithmetische Sätze gibt, die aus diesem System nicht abgeleitet werden können (Krämer 1988: 146). Unter anderem bedeutet dies, daß es keinen Algorithmus gibt, der als Eingabe irgendeine Aussage über die natürlichen Zahlen enthält und dessen Ausgabe feststellt, ob diese Aussage wahr oder falsch ist. Was Gödel für den Bereich der Arithmetik gelang, wies Church 1936 für die Prädikatenlogik nach: es gibt keinen Algorithmus, der entscheiden kann, ob ein beliebiger Satz ein Theorem der Prädikatenlogik ist oder nicht (Unentscheidbarkeit der Prädikatenlogik). Ebenso gibt es keinen Algorithmus, der für jedes Computerprogramm entscheiden könnte, ob es jemals zum Halten kommt oder nicht (Halteproblem). Beispiele für weitere nicht-berechenbare Probleme finden sich in Goldschlager/Lister (1990). Die Frage, welche Probleme durch ein algorithmisches Verfahren berechnet werden können, wird in der Informatik in der Theorie der Berechenbarkeit behandelt.
- Ist bekannt, daß es für ein Problem einen Algorithmus gibt, so ist nicht gesagt, daß dieser auch praktisch verwendbar ist, da die Ausführung infolge hoher Zeit- und Speicherressourcen realistischerweise oft nicht möglich ist. Ein effektives Verfahren kann eine Berechnung im Prinzip zwar durchführen können, dafür aber so lange brauchen, daß es praktisch wertlos ist. Mit diesen Fragen befaßt sich die

Komplexitätstheorie. In unserem Kontext ist diese Fragestellung relevant bei der Berechnung von Cliquen (vgl. Kap. 9.4.3.1, S.210ff.).

- Drittens - und dies dürfte in den Human- und Sozialwissenschaften der wichtigste Punkt sein - kann etwas nicht in ein effektives Verfahren gebracht werden, weil es (noch) nicht verstanden wird. Intuition läßt sich schlecht algorithmisieren (Weizenbaum 1978). Für eine Vielzahl von Problemen aus den Human- und Sozialwissenschaften dürfte mangelndes Verständnis der entscheidende Hinderungsgrund sein, sie einem algorithmischen Computermodell zuführen zu können. Eine Formalisierung, wie in Kap. 1.3 erwähnt, ist dabei ein fundamentaler Schritt zum Verständnis eines Problems, "denn immer, wenn eine Problemlösung formalisierbar ist, und immer, wenn sie mechanisierbar ist, existiert ein Algorithmus, dessen Abarbeitung die Problemlösung ergibt" (Krämer 1988: 139).

Abb. 2.7 zeigt in einer bildlichen Darstellung, wie verschwindend klein die berechen- *und* durchführbaren Aufgabenstellungen im Vergleich zu berechenbaren Aufgabenstellungen und dem Universum aller möglichen Aufgabenstellungen sind.

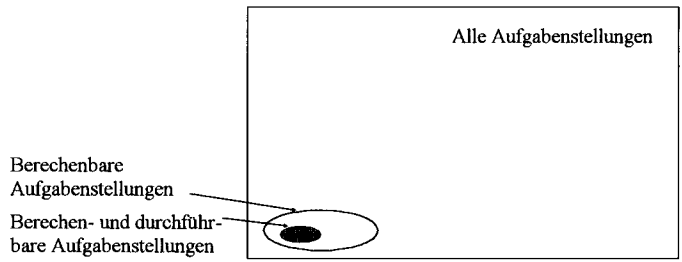


Abb. 2.7: Das Universum der Aufgabenstellungen (Goldschlager/Lister 1990: 94)

Kann ein Vorgang nicht durch einen Algorithmus beschrieben werden, dann gibt es somit keine Chance, ihn jemals durch einen Computer ausführen zu lassen - egal, wie groß oder schnell er ist. Die Rolle von Algorithmen ist damit grundlegend: ohne Algorithmus gibt es kein Programm und ohne Programm gibt es nichts auszuführen.

2.3 Programmiersprachen und Modellierungssoftware

Die Durchführung eines Prozesses auf einem Computer erfordert, daß

- ein Algorithmus entworfen wird, der beschreibt, wie der Prozeß auszuführen ist,
- der Algorithmus als Programm in einer Programmiersprache formuliert wird,
- der Computer das Programm ausführt (Goldschlager/Lister 1990: 20).

Programmiersprachen sind Sprachen zur Spezifikation effektiver Verfahren. Eine Programmiersprache ist ein formales Notationssystem zur Beschreibung von Algorithmen, die von einem Computer ausgeführt werden sollen (Ghezzi/Jazayeri 1989: 375), und ein Programm kann als Implementation eines Algorithmus in einer bestimmten Programmiersprache angesehen werden (Winston 1984: 20).

Eine Programmiersprache ist prinzipiell

- eine formale Sprache (wie z.B. die der Prädikatenlogik 1. Stufe) und
- ein Medium, mit dem Computern Befehle erteilt werden können.

Programmiersprachen dienen also

- zum einen Menschen, die Programme schreiben und lesen, zur Beschreibung von Problemen,
- zum andern Computern, die Programme ausführen und Berechnungen durchführen.

Programmiersprachen lassen sich verschiedenen *Ebenen* zuordnen. Unmittelbar auf dem Computer ausführen lassen sich nur Algorithmen in *Maschinencode*, verfaßt als Folge von 0-1-Tupeln. Befehle auf dieser untersten Ebene der Maschinensprache werden direkt vom Computer abgearbeitet, haben aber den Nachteil, daß sie schwer lesbar und hardwareabhängig sind. Man ging deshalb dazu über, in sog. niederen Programmier- oder *Assembler-Sprachen* die 0-1-Befehle durch mnemotechnische Buchstabenkombinationen abzukürzen (z.B. 10101010 durch ADD) und schließlich ganze Befehlsfolgen in einer einzigen Anweisung zusammenzufassen. Auf diese Weise entstand im historischen Verlauf eine Familie von *höheren oder problemorientierten Programmiersprachen,* die hardwareunabhängig und leicht lesbar sind. Der in diesen Sprachen erstellte Programmcode kann aber nicht unmittelbar maschinell ausgeführt werden, sondern muß mit einem eigenen Programm - Interpreter oder Compiler - in Maschineninstruktionen übersetzt werden.

Da jeder Programmtext in Maschinencode transferiert werden muß, läßt sich alles, was in einer höheren Programmiersprache formulierbar ist, prinzipiell auch auf Maschinenebene formulieren. Höhere Programmiersprachen machen den Computer nicht leistungsfähiger, aber für Menschen besser handhabbar. Alle Programmiersprachen sind also insofern *äquivalent*, als jede algorithmisch lösbare Aufgabe im Prinzip in jeder Sprache gelöst werden kann. Sie sind insofern *unterschiedlich*, als sich bestimmte Probleme mit bestimmten Sprachen besser lösen lassen als mit anderen.

Höhere Programmiersprachen faßt man in Sprachfamilien zusammen. Jeder Sprachfamilie kann eine bestimmte virtuelle Maschine zugeordnet werden, die aus einer bestehenden Allzweckmaschine eine neue, spezielle Maschine entwirft. Eine virtuelle oder gedachte Maschine schottet den Programmierer wie eine Schale von der Maschinenebene ab und es wird der Eindruck erzeugt, als "verstünde" die Maschine die Programmiersprache.[19] Auf programmiersprachlicher Ebene übernimmt diese Abschottung der Interpreter oder Compiler, der den Programmcode in Maschineninstruktionen übersetzt.

Die wichtigsten Sprachfamilien sind imperative, funktionale und logische Sprachen. Bei den *imperativen Sprachen* entsprechen Konzepte der Programmiersprache deutlich Maschinenoperationen und dem Turing-Maschinen-Modell. Ein Programm in einer imperativen Programmiersprache besteht aus einer Folge von sequentiellen Anweisungen an den Computer. Typische imperative Sprachen sind FORTRAN,

[19] Im weiteren Sinn versteht man unter einer virtuellen Maschine jede durch Software hervorgerufene Funktionsänderung einer Rechenmaschine. Ein Schachprogramm ändert z.B. eine reale Maschine in die virtuelle Maschine "Schachautomat" um, die sich wie ein Schachspieler verhält (Ludewig 1985: 28).

PASCAL, ADA, ALGOL und C. Bei *funktionalen Programmiersprachen* werden Programme als Funktionen von Mengen von Eingabewerten in Mengen von Ausgabewerten betrachtet (Claus 1986: 461). Komplexere Funktionen werden durch Zusammensetzung aus einfachen Funktionen gebildet, wobei eine bestimmte Menge an Grundfunktionen vorgegeben ist. Dem Idealbild einer funktionalen Sprache kommt LISP am nächsten (Winston/Horn 1987). Bei *prädikativen oder logischen Programmiersprachen* wird Programmieren als Beweisen in einem System von Tatsachen und Schlußfolgerungen aufgefaßt, und die Problembeschreibung erfolgt in einem logischen Formalismus (Claus 1986: 461). Logische Programmiersprachen realisieren die höchste Abstraktionsebene, was bei den üblichen von-Neumann-Rechnern auf Kosten der Effizienz geht. Eine typische logische Programmiersprache ist PROLOG (Clocksin/Mellish 1987).

Jede Programmiersprache ist im Hinblick auf bestimmte Problemstellungen entwickelt worden. FORTRAN wurde beispielsweise für technisch-naturwissenschaftliche Aufgaben konzipiert und PASCAL für Ausbildungszwecke. Im allgemeinen sind imperative Sprachen eher numerisch orientiert, d.h. sie eignen sich in erster Linie für Operationen mit Zahlen. PROLOG und LISP werden hingegen eher für nicht-numerische Problemstellungen eingesetzt, bei denen weniger Zahlen als Symbole und Zeichenketten manipuliert werden. Ein Überblick über die verschiedenen Sprachfamilien und Einsatzmöglichkeiten findet sich in Goldschager/Lister (1990) und Ghezzi/Jazayeri (1989).

Die naive Annahme, Computerprogrammierung bestehe im Schreiben sequentieller Anweisungen (wie sie in der Simulationsliteratur weit verbreitet ist, z.B. Apter 1971, Dörner 1984), ist falsch. Diese Auffassung trifft nur auf das imperative Modell zu, welches allerdings im allgemeinen und bei den Computermodellierern im besonderen das populärste ist. Richtig ist vielmehr, daß für jedes Programmierproblem ein bestimmtes *Verarbeitungsmodell* zugrunde gelegt wird, die Idee einer bestimmten Maschine, nach deren Vorstellung wir Algorithmen entwerfen. "Dieses Verarbeitungsmodell bestimmt die Konstruktion des Algorithmus und damit auch die sprachlichen Erfordernisse, Formen und Strukturen für seine Implementierung. Es prägt den Stil, in dem das Programm geschrieben ist, den Programmierstil" (Görz 1988: 73).[20] Die *Vorstellung* von einem Verarbeitungsmodell begründet also den Programmierstil und dieser kann durch Programmiersprachen unterstützt bzw. behindert werden (Stoyan 1988: 21).

Der Aussage von Dörner (1984: 346), nach der die Charakteristik einer Sprache nur eine geringe Rolle für die Modellkonstruktion spielt und man bei genügendem Programmiergeschick mit jeder Sprache alles machen kann, können wir nicht zustimmen. Zwar kann man "im Prinzip", wie oben erwähnt, mit jeder Sprache alles machen, aber es gibt für die Modellierung im allgemeinen sowie für bestimmte Problemstellungen im besonderen geeignete und weniger bis völlig ungeeignete Sprachen. Ein

[20] Das Verarbeitungsmodell ist nicht unbedingt mit einer bestimmten Sprachfamilie verknüpft. Hat man z.B. die Vorstellung, daß Algorithmen immer sequentielle Anweisungen an die Maschine sind, so wird man in LISP oder PROLOG anweisungsorientiert programmieren wie in FORTRAN oder C - was durchaus möglich ist. Allerdings ist dies schlechter Programmierstil, da diesen Sprachen eine andere "Philosophie" unterliegt (Stoyan 1988).

sozialpsychologisches Modell in Maschinensprache zu implementieren scheint ebensowenig angebracht wie ein quantitatives Modell, das hochkomplexe numerisch-statistische Verfahren verwendet, in einer symbolisch orientierten Sprache zu formulieren. Ghezzi/Jazayeri (1989: 25-27) stellen Forderungen an Programmiersprachen auf wie Programmierbarkeit (Probleme in angemessener Weise formulieren), Einfachheit und Lesbarkeit, Strukturierung, Effizienz und Portabilität (Übertragbarkeit auf andere Rechnerfamilien). Für die Computermodellierung haben diese Forderungen unterschiedliches Gewicht. Effizienz spielt hier z.B. eine geringere Rolle als in kommerziellen Anwendungen, da die Modelle in der Regel nicht zeitkritisch sind. Hingegen haben Lesbarkeit, Strukturierung und Portabilität eine größere Bedeutung. Maschinennahe Sprachen sind damit völlig ungeeignete Modellsprachen, da sie zwar effizient sind, aber Lesbarkeit, Ausdruckskraft und Portabilität nicht hinreichend gegeben sind.

Höhere Programmiersprachen erlauben eine Programmierung, die sich nahe an die Alltagssprache anlehnt (Schnell 1990: 117), und sind grundsätzlich zu bevorzugen. Von den gebräuchlichen imperativen Sprachen C, FORTRAN, BASIC und PASCAL ist C populär, effizient und hardwarenah, allerdings schwer lesbar und schwierig zu lernen. C hat bislang keine große Basis als sozialwissenschaftliches Modellierungswerkzeug, obwohl es unter professionellen Programmierern die bevorzugte Sprache ist. FORTRAN hat als quantitativ ausgerichtetes Modellierungswerkzeug in den Sozialwissenschaften eine gewisse Tradition und Existenzberechtigung. Aufgrund der mangelnden Strukturierungsmöglichkeit (goto-Sprungbefehl) und der damit verknüpften schweren Lesbarkeit ist FORTRAN aber eher zu meiden (was für neuere Versionen nicht mehr in dem Maße gilt).

Aufgrund ihrer Einfachheit und Ausdruckskraft erscheint PASCAL als ideale imperative Programmiersprache: es unterstützt die disziplinierte Programmierung, ist leicht lesbar und lernbar sowie weit verbreitet. Hingegen mangelt es PASCAL an gewissen numerischen Fähigkeiten. Schnell (1990) tritt insbesondere für PASCAL als Allzweck-Sprache ein: "Die elementaren Grundlagen einer Sprache wie PASCAL, die notwendig sind, um einfache Programme lesen zu können, sind in sehr kurzer Zeit erlernbar. Die notwendige Lerndauer liegt mit Sicherheit unterhalb der entsprechenden Dauer für den Erwerb der Fähigkeiten für das Verständnis von Differentialgleichungen oder phänomenologischen Texten" (Schnell 1990: 116-117).

Von den nicht-imperativen Sprachen sind LISP und PROLOG die gebräuchlichsten. Beide Sprachen werden in der nicht-numerischen, "qualitativen" Modellierung bevorzugt. LISP wurde z.B. in der theoretischen Biologie zur Modellierung von Sozialbeziehungen von Hummeln verwendet (Hogeweg/Hesper 1985), PROLOG bei der Modellierung von Schemata-Konzepten zur Erklärung sozialer Strukturen (Banerjee 1986). In PROLOG muß i.a. weniger Code geschrieben werden als in LISP, und die Konzentration auf die Natur des Problems scheint eher möglich zu sein. PROLOG-Code ist zudem auch für Nicht-Programmierer leichter lesbar als LISP-Code: "vermutlich kann auch jemand, der Prolog nicht kennt, die Bedeutung des Prolog-Textes 'erraten', während dies bei der Lösung in Lisp kaum möglich ist" (Schnupp/Nguyen Huu 1987: 4). Die Bevorzugung von PROLOG vor LISP hat aber noch andere,

wichtigere Gründe, auf die wir später eingehen. Die eben aufgestellten Behauptungen können leicht anhand des einfachen - allerdings numerischen - Beispiels in Abb. 2.8 verifiziert werden.

```
/* PROLOG */

bevoelkerung(usa, 203).
bevoelkerung(indien, 548).
flaeche(usa,3).
flaeche(indien, 548).

bev_dichte(Land, Dichte) :-
   bevoelkerung(Land, Bev),
   flaeche(Land, Fl),
   Dichte is Bev/Fl.

/* LISP */

(defun bevoelkerung land
   (cond ((eq land 'usa) 203)
         ((eq land 'indien) 548) ))

(defun flaeche land
   (cond ((eq land 'usa) 3)
         ((eq land 'indien) 3) ))

(defun bev-dichte land
   (div (bevoelkerung land) (flaeche land) ))
```

Abb. 2.8: Vergleich von PROLOG und LISP anhand des einfachen Problems der Ermittlung der Bevölkerungsdichte (modifiziert nach Schnupp/Nguyen Huu 1987: 3)

Neben Programmiersprachen werden in der Computermodellierung oft höhere Modellierungswerkzeuge eingesetzt, die bestimmte Hilfsmittel bereitstellen und/oder auf spezielle Anwendungsprobleme zugeschnitten sind. Es gibt eine große Vielfalt solcher Werkzeuge, eine Einordnung und Bewertung findet sich z.B. in Zeigler (1976) oder Kreutzer (1986).

Möhring (1990: 20-21) unterscheidet vier verschiedene Typen von Modellierungssystemen.

1. Universelle Programmiersysteme sind die eben angesprochenen Programmiersprachen (PASCAL, C, FORTRAN etc.), bei denen der Benutzer *keine Unterstützung* bezüglich des speziellen Anwendungszwecks erhält. Dafür kann er flexible, an die eigenen Erfordernisse angepaßte Simulationsprogramme erstellen.
2. Simulationsorientierte Simulationssysteme (z.B. SIMULA, SIMSCRIPT, GASP) unterscheiden sich in ihrer Mächtigkeit kaum von allgemeinen Programmiersprachen, bieten dem Benutzer aber Unterstützung für die Erstellung von Simulationsprogrammen, wie etwa Zufallszahlengeneratoren.
3. Modellierungsorientierte Simulationssysteme (z.B. GPSS, DYNAMO, CSMP) stellen bestimmte Modellkomponenten bereits zur Verfügung (z.B. Berechnung von Ableitungen, Integralfunktionen). Da immer nur bestimmte Modellklassen unterstützt werden, sind die Modellbeschreibungsmöglichkeiten hierbei eingeschränkt.

4. Anwendungsorientierte Simulationssysteme (z.B. RAILSIM, HOSPSIM,XCERT) zeichnen sich durch einen zusätzlichen Anwendungsbezug aus und schränken die Einsatzmöglichkeiten weiter ein. In diesen Systemen wird die entsprechende Fachterminologie berücksichtigt bis hin zur "naturgetreuen" Abbildung und Visualisierung von Simulationsabläufen per Computeranimation.

Von (1) nach (4) bieten die Modellierungssysteme mehr und mehr Unterstützung und der Modellierungsaufwand nimmt ab, weil Mittel zur Verfügung gestellt werden, die zu übersichtlicheren und kürzeren Modellbeschreibungen führen. Gleichzeitig sinkt die Modellierungsflexibilität, d.h. die Möglichkeiten der Modellierung insgesamt werden mehr und mehr eingeschränkt.

Aufgrund ihrer Komplexität und der Vielfalt ihres Untersuchungsgegenstandes fordert Möhring (1990: 21) für die Sozialwissenschaften Werkzeuge mit einem breit gefächerten Modellierungsinstrumentarium, gleichzeitig aber auch größtmögliche Problemnähe des Beschreibungsformalismus. Möhring entwickelt eine eigene funktionale Modellbeschreibungssprache MIMOSE, deren Vorzug gegenüber klassischen Programmiersprachen allerdings nicht unmittelbar einleuchtet.

Solche Simulationssysteme werden in den Sozialwissenschaften - mit Ausnahme von DYNAMO - auch kaum genutzt. Vielmehr sind die meisten neueren sozialwissenschaftlichen Simulationen in allgemein verfügbaren Hochsprachen wie FORTRAN, PASCAL, LISP oder PROLOG geschrieben (Schnell 1990: 116). Diese Sprachen sind flexibel und allgemein genug für Modellierungszwecke, werden - im Gegensatz zu speziellen Simulationswerkzeugen - von einem breiten, formal geschulten Benutzerkreis verstanden und sind leicht zu lernen. Ein wichtiger Grund für die Verwendung von Hochsprachen liegt darin, daß es kein Simulationsprogramm geben kann, das nicht mittels der Hochsprachen realisiert werden kann, während die Spezialsoftware die Klasse möglicher Modelle zu stark einschränkt (Schnell 1990: 116). Das bei höheren Programmiersprachen fehlende Modellierungsinstrumentarium läßt sich eventuell durch Programm-Bibliotheken und Hilfsprogramme ersetzen. Schnell (1990) weist z.B. auf eine vollständige DYNAMO-Bibliothek in PASCAL und auf eine Sammlung von PASCAL-Hilfsprogrammen für eine große Zahl unterschiedlicher Simulationstypen hin.

Im Gegensatz zu Möhring (1990) wird hier also die Meinung vertreten, daß die üblichen höheren Programmiersprachen ideale Modellierungssprachen sind und jeder formal geschulte Sozialwissenschaftler

- in der Lage sein müßte, eine solche zu erlernen und
- diese als Modellierungssprache einzusetzen.

Abschließend soll noch auf einen Aspekt hingewiesen werden, der im Lauf dieses Buchs immer wieder relevant wird und der den *Aufbau* von Computerprogrammen betrifft. Moderne Computerprogramme können als formale Texte betrachtet werden, die in hierarchisch strukturierte Einzelteile zerfallen. Die hierarchischen Strukturen sind eine Folge der schrittweisen Verfeinerung von Problemlösungen, was in der Informatik mit "teile-und-herrsche" beschrieben wird. "Der Grundgedanke ist, den auszuführenden Prozeß in zahlreiche Einzelschritte zu zerlegen, von denen jeder durch einen Algorithmus beschrieben werden kann, der weniger umfangreich und ein-

facher ist, als der für den ursprünglichen ganzen Prozeß" (Goldschlager/Lister 1990: 31-32). Auf diese Weise zerfällt ein Programm in *hierarchische Programmschichten*, so daß es einfache Teilprogramme (Prozeduren) gibt, welche primitive Operationen auf einer niedrigen Ebene ausführen und Prozeduren auf einer höheren Ebene, die die einfachen Prozeduren verwenden. Mit anderen Worten: Teilprogramme oder Prozeduren lösen bestimmte Aufgaben, wobei diese Teilprogramme wiederum Teilprogramme benutzen, die noch elementarere Aufgaben lösen usf. Dieses Aufbauprinzip ist in der Computerwissenschaft Ausdruck der bedeutenden Grundidee der *Prozedurabstraktion*, nach der der Prozeß der Erzeugung neuer Prozeduren durch Kombination bereits bestehender erfolgt. Prozedurabstraktion befreit also den Modellierer von ablenkenden Details, indem vorhandene Prozeduren verwendet werden (Winston/Horn 1987: 45).

2.4 Klassifikationsschemata von Computermodellen

Bei der Klassifikation von Computermodellen hält man sich in der Literatur meist an eines der folgenden Prinzipien. Die Modelle werden entweder inhaltlich nach dem Gegenstands- bzw. Anwendungsbereich geordnet oder aber mit Hilfe eines Schemas nach formalen Merkmalen. Seinem klassischen Überblicksartikel "Simulation of Social Behavior" legt Abelson (1968) z.B. folgende einfache inhaltliche Klassifikation zugrunde.

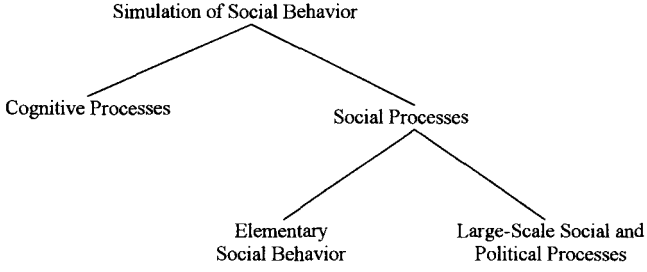


Abb. 2.9: Klassifikation sozialwissenschaftlicher Computermodelle nach inhaltlichen Kriterien (nach Abelson 1968)

Informationshaltiger und brauchbarer erscheinen formale Ordnungsschemata. Ein solches Klassifikationsschema nach Merkmalsklassen wie Verwendungszweck, Zeitabhängigkeit etc. liefert Troitzsch (1990: 12-21). Wir geben das Schema in der gerafften Form von Möhring (1990: 9-11) wieder, die Merkmalsklasse Bildbereich wurde in Tab. 1.1 S.13 bereits genannt.

Tab. 2.1: Klassifikation von Computermodellen nach formalen Kriterien (Möhring 1990)

Verwendungszweck	
Deskription:	Beschreibung von Zusammenhängen
Prognose:	Ableitung zukünftiger möglicher Entwicklungen
Entscheidung:	Ableitung von Prognosen durch Ausprobieren und Bewertung unterschiedlicher Strategien
Didaktik:	Vermittlung wissenschaftlicher Erkenntnisse
Zeitabhängigkeit	
Statisch:	Unabhängigkeit der Modelle von der Zeit
Dynamisch:	Zeitliche Veränderbarkeit von Modellstruktur und Modellverhalten
Zeitstruktur	
Diskret:	Eintreffen von Modellereignissen zu diskreten Zeitpunkten (z.B. Differenzengleichungen)
Kontinuierlich:	"Fließendes" Eintreffen von Modellereignissen (z.B. Differentialgleichungen)
Grad der Determiniertheit	
Deterministisch:	Präzise Vorhersage von Modellereignissen
Stochastisch:	Wahrscheinlichkeitsaussagen zu zukünftigen Modellereignissen
Allgemeine Modellstruktur	
Makromodell:	Modelle bestehen aus einem Element
Mikro-/Mehrebenenmodell:	Modelle enthalten mehrere Elemente auf verschiedenen Ebenen
Allgemeine Struktur des Modellverhaltens	
Linear:	Beziehungen zwischen Modellelementen sind durch lineare Funktionen beschreibbar
Nichtlinear:	Beziehungen zwischen Modellelementen sind durch allgemeine Funktionen beschreibbar
Wertebereich der Merkmale von Modellelementen	
Qualitativ:	Formalisierung qualitativer Eigenschaften durch nominal- bzw. ordinalskalierte Wertebereiche
Quantitativ:	Ratioskalierte Wertebereiche

In dieses Schema eingeordnet entsprechen z.B. die im zweiten Teil behandelten D-H-L-Modelle dynamischen, deterministischen, qualitativen Mehrebenenmodellen zur Prognose. Dies bedeutet mit anderen Worten, daß in diesen Modellen die Entwicklung abhängig ist von der Zeit, Wahrscheinlichkeiten keine Rolle spielen, qualitative Merkmale abgebildet werden und Modelleigenschaften auf mehreren Ebenen betrachtet werden.

2.5 Zum Nutzen von Computermodellen

In der Literatur wird eine Vielzahl von Argumenten angeführt, die den positiven Einfluß der Verwendung von Computern auf die Theorien- und Modellbildung in den empirischen Wissenschaften explizit machen. Das folgende Schema versucht die wichtigsten Argumente in vier Punkten zusammenzufassen.[21]

1. Formalisierung *und* Ausführbarkeit

   Da Programmiersprachen exakt definierte Kunstsprachen sind, bilden Programmiersprachen im Prinzip formale Sprachen wie die von Differentialgleichungen, Prädikatenlogik oder Graphentheorie. *Computermodelle* besitzen folglich die *Eigenschaften formal-mathematischer Modelle* und unterstützen damit den obengenannten Zwang zur Explikation von Theorien. Wie andere Formalisierungsformen ist Computermodellierung ein Mittel der Begriffsklärung, da in Computermodellen Begriffe ebenso präzisiert, einheitlich verwendet und Zusammenhänge explizit beschrieben werden müssen. Programmiersprachen haben aber im Vergleich zu anderen formalen Sprachen darüber hinaus den Vorteil, daß der *formale Text von der Maschine bearbeitet* werden kann, so daß das Verhalten des Rechners von dem Modell determiniert wird. Während ein verbales oder formales, nicht computerisiertes Modell eine abstrakte, undynamische Struktur darstellt, "macht" das Computermodell etwas: "Ein Simulationsmodell 'verhält sich'; es ist etwas Konkretes und lenkt unmittelbar den Blick auf merkwürdige, falsche oder unvollständige Konzeptualisierungen" (Dörner 1984: 346). Im Fall der Repräsentation theoretischer Aussagen führt der Rechner zu einer "Dynamisierung theoretischer Aussagen" (Harbordt 1974: 262); er zeigt das "Verhalten" der Theorie: "When that computer is programmed to represent one's theory, its processes are then synonymous with those of the theory, and pressing the GO button sets the *theory* in motion. Its processes begin literally to move, their consequences interact, evolve something new, and thus realize the theory's extended consequences. It makes a theory 'go somewhere'" (McPhee 1960: 4, Hervorhebung McPhee).

2. Deduktive Mächtigkeit

   Einer der zwingendsten Gründe für die Anwendung von Computern ist der, daß sie die blinden Flecke in unserem Denken bloßlegen (Weizenbaum 1978: 96). Im Gegensatz zum Menschen ist beim Computer die Ableitungsfähigkeit nicht beschränkt. Die Simulation ermöglicht schnelle und korrekte Ableitungen auch aus einem komplexen, für Menschen nur mehr schwer oder überhaupt nicht mehr überschaubaren Aussagensystem (Harbordt 1974: 270). Programme lassen sich durch eine Fülle von Daten nicht verwirren, sondern führen *vollständige* und *korrekte* Ableitungen durch. Da ein Simulationsmodell nur ablauffähig ist, wenn alle benötigten Informationen bereitgestellt werden, kann das Explizit-machen-müssen theoretischer Annahmen die Freilegung von Wissenslücken zur Folge haben. Beispielsweise läßt sich dadurch entdecken, daß bestimmte theoretische Konsequenzen nur

[21] Vgl. zum folgenden insbesondere Gullahorn/Gullahorn (1965), Apter (1971), Ziegler (1972), Harbordt (1974), Dörner (1984), Lischka/Diederich (1987), Schnell (1990).

bei Vorliegen bestimmter - bislang nicht explizit gemachter - Annahmen auftreten. Sylvan/Glassner (1985) entdeckten z.B., daß die Konflikt-Kohäsions-Behauptung von Simmel nicht aus seinen Annahmen ableitbar ist (vgl. Kap. 4.3.2). Umgekehrt kann die Durchführung einer Simulation zur Entdeckung *unerwarteter* Effekte führen, wie sie etwa Schnell (1990) anhand einer Arbeit über die Bedeutung sozialer Netzwerke für die Entstehung kollektiver Bewegungen schildert.

3. Experimenteller Charakter

   Die vordergründigste - und in den genannten Punkten bereits angeklungene - Eigenschaft von Computermodellen ist die ungehinderte Möglichkeit zum Experimentieren. Der Computer kann insofern ein neues Verhältnis zu Theorien eröffnen, als "er die Möglichkeit bietet, nicht nur natürliche, sondern auch künstliche Realisationen eines theoretischen Konzepts in ihrem Verhalten zu untersuchen und so Theorien zusätzlichen Prüfungen unterziehen zu können" (Kobsa 1986: 123-124). Dies ist besonders wichtig in Wissenschaftsbereichen, die sich aus ethischen, physischen oder sonstigen Gründen der experimentellen Methode verschließen. Computermodelle ermöglichen beliebige Experimente, indem entweder die Inputs oder das Programm selbst (z.B. bestimmte Zusammenhänge) variiert und die Folgen beobachtet werden. Der experimentelle Charakter von Computermodellen hat im einzelnen etwa folgende positive Auswirkungen:
   - Alternative Annahmen/Hypothesen und deren Konsequenzen können durchgespielt werden.
   - Modelle können z.B. in allen Einzelheiten gleich gemacht werden außer in dem kritischen Aspekt, unter dem die Modelle verglichen werden sollen (Lindenberg 1971: 100).
   - Der Gültigkeitsbereich eines Modells kann getestet werden, indem man Extrembedingungen untersucht (Lischka/Diederich 1987: 28).
   - Die Robustheit des Modells und die Bedeutung einzelner Komponenten kann erfaßt werden, indem man Teile entfernt oder hinzufügt (Lischka/Diederich 1987: 28).
   - Unbegrenzte "mögliche Welten" und "What-would-be-if"-Szenarios lassen sich realisieren.

4. Universalität

   Kobsa (1986: 123) weist darauf hin, daß es derzeit kaum "Unmöglichkeitsbeweise" gibt, also theoretische Grenzen der Anwendung des Computers als Hilfsmittel bei der Theoriebildung. Einzige Ausnahme dürfte dabei das obengenannte Gödelsche Unvollständigkeitstheorem sein, dessen Auswirkung auf das "High-Level-Verhalten" des Computers aber noch sehr unklar ist. Verschiedene Computertheoretiker setzen sich z.B. mit der Frage auseinander, ob das Gödel-Theorem intelligente Maschinen prinzipiell unmöglich macht (vgl. hierzu die Literaturhinweise in Kobsa 1986). Solche Fragen sind in unserem Kontext aber nicht weiter relevant. Aus unserer Sicht ist ein Computer ein Allzweck-Werkzeug, das prinzipiell programmiert werden kann, um das Verhalten jedes anderen Systems nachzumachen (Gullahorn/Gullahorn 1965: 433). Eine prinzipielle Beschränkung ergibt sich weder für bestimmte Wissenschaftsgebiete (Sozial- versus Naturwissen-

schaften), noch für bestimmte Modellarten (z.B. quantitative versus qualitative Modelle).

2.6 Computermodelle in den Sozialwissenschaften

Wir haben uns bisher allgemein mit Modellen und Computermodellen beschäftigt und wollen in diesem Teil den "Stand der Kunst" anhand von typischen Simulationsformen in den Sozialwissenschaften darstellen. Abschließend sollen mit diesen Modellierungsformen verknüpfte Probleme aufgezeigt werden.

In der Vergangenheit haben sich in den Sozialwissenschaften einige Modellbildungsansätze etabliert, die sich grob drei Klassen zuordnen lassen: System-Dynamics-Modelle, Mikrosimulationsmodelle und Mikro-/Makro- oder Mehrebenenmodelle. Sie werden überblicksartig beschrieben in Schnell (1990), Troitzsch (1990) und Möhring (1990). Aktuelle Beispiele für derartige Simulationen finden sich in Kreutz/Bacher (1991).

Der *System-Dynamics-Ansatz* geht auf Forrester zurück (vgl. z.B. Forrester 1971) und wurde für die Entwicklung von Stadt-, Regional- und Weltmodellen verwendet. Die Weltmodelle erregten insbesondere im Zusammenhang mit den Veröffentlichungen des "Club of Rome" große Aufmerksamkeit auch außerhalb der Wissenschaft (Meadows et al. 1972). System-Dynamics-Ansätze lassen sich grob dadurch charakterisieren, daß sie ein System (die Stadt, die Region, die Welt) in eine möglichst kleine Menge von aggregierten Variablen (z.B. Bevölkerungszahl, Produktion) zerlegen, welche Merkmale des Gesamtsystems beschreiben. Der letzte Punkt ist entscheidend: Variablen im System-Dynamics-Ansatz charakterisieren nicht die einzelnen Elemente des Systems, sondern immer das System als Ganzes. Die Relationen, die zwischen den Variablen herrschen, werden durch Gleichungen, meist Differenzen- oder Differentialgleichungen, ausgedrückt. Der Kern eines System-Dynamics-Modells besteht in der Verkettung dieser Gleichungen, welche die Struktur des Systems wiedergeben sollen. Der Ablauf eines solchen Modells gestaltet sich so, daß mit den eingegebenen Anfangswerten die Gleichungen berechnet werden. Auf der Basis der berechneten Werte wird ein neuer Zyklus gestartet, dessen Ergebnis Grundlage für die Berechnung des nächsten Zyklus ist usw. Der Modelloutput besteht aus Zeitreihen in tabellarischer oder grafischer Form. Als Modellbeschreibungssprache dient hierbei meist die oben erwähnte, ebenfalls von Forrester entwickelte Sprache DYNAMO. Eingeordnet in das obige Klassifikationsschema handelt es sich bei diesen Programmen um quantitative, zeitdiskrete, deterministische, nichtlineare, dynamische Makromodelle, die zu Erklärung, Prognose und Entscheidungsunterstützung dienen können.

System-Dynamics-Modelle zeigen eine Reihe von eklatanten Mängeln. Kritisiert wurde inbesondere die extrem hohe Aggregierung der Variablen (beim Weltmodell etwa: Bevölkerungszahl, Umweltverschmutzung etc.) und die mangelnde empirische Verankerung und Validierung. Eine ausführliche Kritik findet sich bei Harbordt (1974). Trotz neuerer, verbesserter System-Dynamics-Modelle wie z.B. GLOBUS

und einer Abkehr vom Anspruch eines prognostischen Werkzeugs hin zum heuristischen Hilfsmittel wird ein wahrnehmbarer Einfluß dieser Modelle auf die Theoriebildung in den Sozialwissenschaften bestritten (Schnell 1990: 112-113). Eine neuere Einführung in das system-dynamische Konzept mit natur- und sozialwissenschaftlichen Anwendungen findet sich in Bossel (1992).

Die System-Dynamics-Modellierung ist ein typischer *Makroansatz*, bei dem das Verhalten des Systems durch das Zusammenwirken verschiedener (Makro-)Variablen beschrieben wird. Im Gegensatz dazu steht bei *Mikrosimulationsmodellen* das Verhalten von individuellen Einheiten im Vordergrund. Mikrosimulationsmodelle sind stochastische, zeitdiskrete Computersimulationsmodelle auf Mikroebene, die der Prognose und Entscheidungsunterstützung dienen. Diese Modelle bestehen aus einer sehr großen Zahl von individuellen Elementen (z.B. Personen, Haushalte), deren Merkmale durch eine Menge von Variablen beschrieben werden (z.B. Kinderzahl, Alter, Einkommen) und Vorschriften, wie sich diese verändern sollen. Simuliert werden die Veränderungen der Merkmale der Elemente über die Zeit. Der Computer arbeitet Element für Element ab und modifiziert die Merkmale entsprechend den Vorschriften. Sind alle Elemente bearbeitet, ist ein Modellauf abgeschlossen und der nächste Lauf beginnt. Der Modelloutpout besteht aus Statistiken für die interessierenden Variablen (z.B. Änderungen im Altersaufbau einer Bevölkerung), die wie bei einer realen Stichprobe berechnet werden. Anders als bei System-Dynamics-Ansätzen wird hier also immer auf der Ebene der Systemelemente simuliert und anschließend werden statistische Maßzahlen berechnet. Solche Mikrosimulationsmodelle wurden z.B. zur Nachbildung von Meinungsbildungsprozessen oder zur Beurteilung möglicher Folgen von Gesetzesvorhaben verwendet.

Eine Verbindung zwischen Makro- und Mikromodellen stellen die sog. *Mehrebenen-* oder *Mikro-Makromodelle* dar. Mit diesen Modellen wird versucht, Effekte auf der Makroebene durch Vorgänge auf der Mikroebene zu erklären. In der aktuellen Literatur werden diese Modelle als sehr mächtig angesehen. Möhring (1990) bezeichnet diesen Ansatz aus zwei Gründen als den vielversprechendsten für die Sozialwissenschaften. Erstens enthalten Mehrebenenmodelle die obigen Modelltypen als Spezialfälle und verknüpfen sie miteinander. Zweitens unterstützt dieser Modelltyp die Erforschung zentraler Fragestellungen in den Sozialwissenschaften, wie etwa die Untersuchung kollektiver Phänomene, die durch viele Individuen ausgelöst wurden. Schnell (1990) bezeichnet die (Mikro-)Simulation individueller Akteure zur Erklärung von Makrophänomenen als zumindest derzeit einzig fruchtbaren Ansatz.

Mikro-Makrophänomene seien an einem kurzen Beispiel erläutert. Bainbridge (1987) stellt zwölf einfache Simulationsprogramme vor, welche klassische soziologische Fragestellungen und Kontroversen behandeln. Eines dieser Programme, RACE, das sich auf die Arbeiten von Boudon und Schelling (1978) bezieht, modelliert die nicht intendierten Konsequenzen individuellen Handelns am Beispiel von ethnischer Segregation. Auf der unteren Mikroebene wird eine Menge von maximal 64 Individuen dargestellt, wobei jedes Individuum entweder zur blauen oder grünen Rasse gehört. Die Individuen werden auf die Felder eines Schachbrettes zufällig verteilt, welche Häuser einer Stadt symbolisieren. Jedes Individuum hat die Eigenschaft

"Zufriedenheit" oder "Unzufriedenheit", wobei sich diese Eigenschaft aus der Zusammensetzung der Nachbarschaft und der eingestellten Entscheidungsregel ergibt. Eine solche Entscheidungsregel lautet z.B. "Ich bin nur zufrieden, wenn die Mehrheit meiner Nachbarn gleichrassig ist". Ein grünrassiges Individuum mit sechs grünen und zwei blauen Nachbarn wird danach zufrieden sein, ein blaurassiges wird in der gleichen Nachbarschaft unzufrieden sein. Unzufriedene Individuen versuchen in leerstehende Felder umzuziehen, aber nur, wenn dadurch der Zustand Zufriedenheit hergestellt werden kann. In jedem Simulationsschritt erfolgen nun solche Umzüge. Unter bestimmten Bedingungen entstehen nach einigen solchen Schritten aus der ungeordneten, zufälligen Verteilung der Rassenmitglieder zusammenhängende Rassencluster, d.h. es kommt zu ethnischer Segregation.

Dieses einfache Modell illustriert in sehr elementarer Form die nicht beabsichtigten Konsequenzen sozialen Handelns und die Tatsache, daß ein soziales System mehr ist als die Summe der Individuen. Es ist zugleich ein einfaches Beispiel für ein Mehrebenenmodell: auf der Mikroebene werden Individuen mit bestimmten Merkmalen (Rassenzugehörigkeit, Entscheidungsregel) abgebildet. Auf der Makroebene wird das System als Ganzes in ein einziges Objekt transferiert mit den relevanten Objekteigenschaften "ungeordnete Struktur" oder "in bestimmte Rassen segregiert". Der Vorteil dieses Ansatzes ist, daß Makrophänomene (Rassensegregation) durch Verhalten auf Mikroebene (Rasse, Entscheidungsregel) erklärt werden können. Die später behandelten D-H-L-Modelle sind ebenfalls Beispiele für Mikro-Makromodelle.

2.7 Probleme konventioneller Modellierung

Das eben vorgestellte "Modellierungsparadigma" sowie die allgemeine Modellierungsrichtung in der sozialwissenschaftlichen Literatur ist quantitativ orientiert. Harbordt (1974) beschränkt sich weitgehend auf quantitative Modelle, für Mayntz (1967: 23) sind *alle* Simulationsmodelle quantitativ. In quantitativen Modellen ist der Bildbereich ein numerisch ausgerichteter Kalkül und/oder eine numerisch orientierte Programmiersprache. Eine derartige Modellierung zwingt zur vollständigen Abbildung des Urbilds in numerische Konzepte. Wir verwenden für Computerprogramme dieses numerischen Typs im folgenden auch den Ausdruck "konventionelle (Computer-) Modelle". Viele Autoren engen Modellierung weiter auf Systeme von Gleichungen und sogar Differentialgleichungen ein. Beispielsweise definiert Dörner (1984: 345) Simulationsmodelle ausschließlich als Gleichungssysteme: "Ein Simulationsmodell basiert auf einer formalen Systembeschreibung und damit letzten Endes auf einem System von Gleichungen, meist wohl auf einem System von Differentialgleichungen". Analog behandelt Troitzsch (1990) in seinem Buch "Modellbildung und Simulation in den Sozialwissenschaften" ausschließlich Modelle, die auf Differentialgleichungen beruhen.

Die quantitative Orientierung und Modellierung in strengen, numerisch ausgerichteten Kalkülen dürfte zwei grundlegende Ursachen haben. Ein erster Grund dürfte in dem vorherrschenden physikalistischen Wissenschaftsideal liegen, das mit dem

Standardtheorienkonzept verknüpft ist. Mit der "Ideal-Wissenschaft" Physik als Richtschnur für die Theorienkonzepte der Logischen Empiristen und Kritischen Rationalisten lautete der implizit und explizit gegebene Ratschlag an die Sozialwissenschaftler, wie die Physiker nach quantitativen Begriffen und Zusammenhängen zu suchen. Die Angabe genauer, quantitativer Funktionsregeln für Zusammenhänge führt im Sinn von Popper dazu, daß eine entsprechende Theorie leichter falsifizierbar und damit empirisch gehaltvoller wird. Ein zweiter Grund dürfte in der tiefen, auch historisch bedingten Verwurzelung konventioneller Computersimulation in numerisch orientierten Modellierungswerkzeugen liegen. Der Computer wurde und wird in den Sozialwissenschaften als Maschine gesehen, die Zahlen verarbeitet. Alternativen zur Sichtweise des Computers als Zahlen- und Arithmetik-Maschine sind nicht bekannt oder werden als nicht relevant erachtet. Unglücklicherweise sind mit dem physikalistischen Wissenschaftsideal und der restriktiv-einseitigen Benutzung des Computers einige wesentliche Nachteile verbunden, die Nutzen und Einsatzmöglichkeiten sozialwissenschaftlicher Computermodelle empfindlich mindern.

Erstens setzen quantitative Modelle metrische Begriffe voraus, die in den Sozialwissenschaften selten gegeben sind. Für die Verwendung in solchen Modellen müssen nicht-metrische Begriffe, wie z.B. klassifikatorische Terme, in kontinuierliche metrische Variablen überführt werden. In Einzelfällen kann dies gelingen (vgl. z.B. die Transformation von ordinalen Präferenzen in kardinale Nutzenfunktionen), viele derartige Versuche scheinen aber zumindest fragwürdig.

Zweitens setzen quantitative Modelle die genaue numerische Angabe von Zusammenhängen voraus. In den Sozialwissenschaften kann aber die exakte quantitative Form einer Funktion oft nicht bestimmt werden. Vielmehr liegen Zusammenhänge und Gesetzmäßigkeiten oft nur qualitativ vor. Qualitative Schemata von Gesetzmäßigkeiten, wie sie typischerweise in den Sozialwissenschaften auftreten, sind z.B. die folgenden Wenn-dann-Aussagen (vgl. z.B. Ziegler 1972: 219):

wenn x dann y,
wenn x dann nicht y,
je größer x desto größer y,
wenn x größer als 0 dann ist y kleiner als z,
wenn x und y zunehmen, dann nimmt auch z zu.

In diesen schematischen qualitativen Aussagen werden zwar Zusammenhänge angegeben, aber nicht in Form der Nennung einer genauen quantitativen Funktionsregel. Zahlen können in qualitativen Gesetzen vorkommen, der präzise Zusammenhang wird aber nicht spezifiziert. Ein konkretes Beispiel dafür, daß das grundlegende Gesetz nur in qualitativer Form angegeben werden kann, sind die unten vorgestellten Balancetheorien.

Wenn die fundamentalen Beziehungen zwischen den Variablen nicht genau formuliert werden können, sind die in der Physik üblichen Modelle mit Differentialgleichungen, Integralgleichungen oder Funktionalanalysis nicht anwendbar (Rapoport 1980: 27). Man behilft sich in diesen Fällen deshalb oft damit, daß man qualitative Zusammenhänge künstlich quantifiziert und "präzisiert". Die "Präzisierung" vager Zusammenhänge durch numerische Funktionen ist bei konventionellen Modellen eine

weit verbreitete "Methode". Als Beispiel sei auf die eben vorgestellten System-Dynamics-Modelle verwiesen. Diese Modellierungsmethode macht häufig von sog. Tabellenfunktionen Gebrauch. Tabellenfunktionen werden konstruiert, indem man die *qualitativen* Kenntnisse über die Beziehung von z.B. zwei Variablen in eine grafische Darstellung in einem cartesischen Koordinatensystem grob umsetzt, Wertepaare aus der Grafikdarstellung abliest, diese in Tabellenform bringt und die Tabellenfunktion anstelle der expliziten Funktion als Ausdruck der Variablenbeziehung im Modell verwendet. Auf diese Weise wird eine Beziehung, die nur qualitativ vorliegt, "quantifiziert" (vgl. Harbordt 1974: 130-131). Derartig gewonnene tabellarische Funktionen sind natürlich ziemlich subjektiv und spiegeln eine nicht vorhandene Präzision vor. Wir halten dieses Vorgehen für erheblich fragwürdig.[22]

Eine Mathematisierung im Sinn der Angabe einer quantitativen Funktionsregel führt zwar, wie oben angedeutet, zu Präzisierungen in dem Sinn, daß die Theorien leichter falsifizierbar und damit empirisch gehaltvoller werden. Allerdings wird dabei der Gehalt häufig so stark erhöht, daß die präzisierte Theorie sich tatsächlich kaum empirisch bewähren kann. Dies dürfte der Grund sein, warum viele Soziologen eine Mathematisierung im quantitativen Sinn ablehnen und Theorien relativ allgemein formulieren, um überhaupt empirisch erfolgreich anwendbar sein zu können (Westermann 1987: 21).

Ein Ausweg aus dem Dilemma, lediglich über qualitative Begriffe und Zusammenhangs-Hypothesen zu verfügen und darauf basierende Modelle ohne künstliche "Präzisierungen" maschinell darstellen zu wollen, sind qualitative oder symbolische Computermodelle. Hierzu gehören z.B. die unten besprochenen Modelle von Gullahorn/Gullahorn (1963) oder Doran (1985). In symbolischen Modellen werden Begriffe und Zusammenhänge normalerweise in ihrer ursprünglichen Verwendung belassen. Beispielsweise können die erwähnten qualitativen Gesetzmäßigkeiten auch so in das Computermodell übernommen werden, ohne den genauen Zusammenhang weiter festzulegen. Aus mathematischer Sicht behandeln diese nicht-quantitativen Modelle "abstrakte Relationen und 'Bündel' solcher Relationen, die Strukturen genannt werden. Modelle, bei denen Strukturen im Mittelpunkt stehen, nennen wir strukturelle Modelle. Quantitäten werden selbstverständlich aus solchen Modellen nicht ausgeschlossen, aber sie dienen gewöhnlich als Indices von Strukturen" (Rapoport 1980: 28). Dieser Modelltyp stellt sein Urbild somit auf nicht-numerische Weise dar, bei dem der Bildbereich eine nicht-quantitativ ausgerichtete formale Sprache ist. Als Medium, in dem qualitative Modelle repräsentiert sein können, genügen in der Regel naive Mengenlehre und Prädikatenlogik erster Stufe. Auf der maschinellen Ebene eignen sich symbolisch orientierte Computersprachen, die in der Lage sind, logische und mengensprachliche Konzepte darzustellen. Wir können diesen Ansatz in Anlehnung an Faulbaum (1986, 1991) auch als sehr "voraussetzungsarme" oder "very soft" Modellierung bezeichnen. Mit diesem Unternehmen der sanften

[22] Diese kritischen Bemerkungen sollen nicht pauschal verstanden werden. In bestimmten Fällen mag die Modellierung sozialer Phänomene mit Differentialgleichungen oder allgemein Gleichungssystemen angebracht sein. Sinnvolle Beispiele finden sich in Rapoport (1980). Als *allgemeines* Modellierungsinstrumentarium ist die Beschränkung auf Gleichungssysteme aber zu eng und zu restriktiv.

Modellierung können zum einen nicht-quantitative, mathematische Strukturmodelle dem Computer zugeführt werden, zum andern müßte sich darin jede allgemeine, verbal formulierte Theorie rekonstruieren und "sanft präzisieren" lassen.

Der qualitative Modelltyp wird in der Literatur kontrovers eingeschätzt. Während in manchen Texten qualitative Modelle nur am Rande erwähnt (z.B. Troitzsch 1990) oder gleich per definitionem ausgeschlossen werden (vgl. die Definition von Dörner), betonen andere wie Schnell (1990: 114) die große Bedeutung solcher Modelle für die Theoriebildung in den Sozialwissenschaften.

Qualitative Modelle beseitigen den Nachteil, daß Begriffe und Zusammenhänge nicht unumschränkt künstlich quantifiziert werden müssen. Trotzdem bleiben auch bei diesem Modelltyp - in dem sich ja nur der Repräsentationsformalismus in einen qualitativen ändert - bestimmte Schwachpunkte erhalten.

- In konventionell aufgebauten - quantitativ *und* qualitativ ausgerichteten - Computerprogrammen bereitet es Schwierigkeiten, das Modell ohne großen Aufwand zu ändern und *bestimmte Modellelemente einfach hinzu- oder wegzunehmen*, was für den Experimentiercharakter entscheidend ist.
- Boden (1984) bemängelt an den konventionellen Computermodellen, daß diese *undurchsichtig* seien und nicht angeben können, aufgrund *welcher* Mechanismen ein bestimmter Output zustande kommt.

Die Ursache für beide Schwachstellen liegt darin, daß in Programmen konventionellen Typs das benutzte Wissen implizit im Programmcode versteckt und nicht explizit codiert ist. Dies hat zur Folge, daß das modellspezifische, inhaltlich relevante Wissen nur schwer von der Steuerung und Bearbeitung des Wissens getrennt werden kann. Wenn die wesentlichen Modellannahmen zwischen unwesentlichen Programmcodes versteckt sind, dann sind sie für Außenstehende nicht zugänglich und damit nicht kritisierbar. Dies hat fatale Konsequenzen für Verständnis und Akzeptanz solcher Modelle. Boden (1984) plädiert deshalb dafür, die Modelle sowohl qualitativ als auch in der Architektur von Expertensystemen zu formalisieren, die eine "durchsichtigere" Modellierung ermöglichen.

2.8 Künstliche Intelligenz

Werden die Forderungen von Boden berücksichtigt, so ergeben sich für die sozialwissenschaftliche Computermodellierung enge Verbindungen mit der Künstlichen Intelligenz (KI, auch AI von "Artificial Intelligence"). Denn erstens beschäftigt sich die KI mit der maschinellen Darstellung nicht-quantitativen, symbolischen Wissens, und zweitens werden die von Boden angesprochenen Expertensysteme der KI zugerechnet. Obwohl grundlegende Allgemeinkenntnisse der KI für unsere Zwecke nicht relevant werden, sind doch einige generelle Bemerkungen zur KI angebracht, damit unbefangene Leser zumindest eine vage Vorstellung davon bekommen, um was es in der KI überhaupt geht.

Künstliche Intelligenz wird im allgemeinen als Teilgebiet der Informatik verortet - allerdings mit starken Bezügen zu den Humanwissenschaften (kognitive Psychologie

und Linguistik). KI ist der Versuch, intelligentes, menschliches Verhalten auf Rechenanlagen nachzubilden. Sie befaßt sich mit Computertechniken zur Lösung von Aufgaben, zu deren Bewältigung Menschen Intelligenz einsetzen (Tanimoto 1990: 24). Das Ziel ist, Computerprogramme zu schreiben, die bestimmte Aspekte menschlichen Verhaltens nachahmen. In der Künstlichen Intelligenz arbeitet man beispielsweise daran, daß Computer die deutsche Sprache verstehen, Bilder erkennen, wichtige Entscheidungen treffen oder einen neurotischen Menschen simulieren.

KI-Forscher verwenden im allgemeinen andere Methoden, Techniken und Werkzeuge als traditionelle Informatiker. Da es in der KI vorrangig um qualitatives Wissen und nicht um Zahlen geht, werden beispielsweise symbolisch orientierte Programmiersprachen wie LISP und PROLOG bevorzugt. Ebenso wurden im Verlauf der historischen Entwicklung neuartige Methoden und Techniken ausgearbeitet, die auf die spezifischen Probleme der KI zugeschnitten sind.

Grundlagen-Forschungsgebiete der KI sind die bereits angedeuteten Wissensrepräsentationsmethoden, (heuristische) Suchverfahren, Planungstechniken, maschinelle Lernverfahren oder automatisches Schließen. Die Wissensrepräsentation wird weiter unten ausführlicher behandelt. Für die empirischen Wissenschaften können in Zukunft, neben den später vorgestellten Expertensystemen, insbesondere die folgenden KI-Anwendungsgebiete relevant werden:

- Automatisches Beweisen (Theorem-Beweisen)
  Beim automatischen Beweisen übernimmt der Computer die Aufgabe eines Mathematikers. Die Maschine beweist mathematische Theoreme und zwar so, daß Mathematiker diesen Beweis als gültigen Beweis akzeptieren. Mit jüngst entwickelten Programmen gelang der Nachweis, daß Computer sogar umfangreiche Theorien über ein begrenztes Gebiet wie die Arithmetik aus wenigen einfachen Axiomen entwickeln können (Waltz 1982: 86).
- Bildverarbeitung (Computer Vision)
  Das Ziel der Bildverarbeitung ist, ein Verständnis visueller Informationen zu gewinnen, wie sie uns im täglichen Leben begegnen. Bei der Bilderkennung wird versucht, die Analyse und Interpretation visueller Information im menschlichen Auge und Gehirn mit dem Computer nachzuahmen. Computer können - mit Fernsehkamaras ausgerüstet - gut genug sehen, um mit bestimmten, begrenzten Welten fertig zu werden. Beispielsweise können aus Zeichnungen geometrischer Körper Schlußfolgerungen darüber gezogen werden, welche Arten von Gegenständen vorliegen, welche Beziehungen zwischen ihnen existieren und welche Gruppen sie bilden (Winston 1987: 27).
- Natürlichsprachliche Systeme
  Natürlichsprachliche Systeme erlauben Kommunikation mit dem Computer in Umgangssprachen wie Deutsch oder Englisch. Darunter versteht man konkret Maschinenübersetzung (Übersetzung von Texten aus einer Sprache in eine andere), Textverstehen (Schlüsse aus Texten ziehen, Zusammenfassungen machen), Texterzeugung (z.B. Texte zu Produktbeschreibungen verfassen) und natürlichsprachliche Schnittstellen (Anfragen an Computersysteme, wie z.B. Datenbanken, in Umgangs-

sprache). Am ausgereiftesten ist das Gebiet der natürlichsprachlichen Schnittstellen vor allem bei Datenbankanfragen.

- Maschinelles Lernen
  Automatisches Lernen ist das Gebiet der KI, das sich auf Verfahren der Selbstoptimierung konzentriert (Tanimoto 1990: 331). Informationsverarbeitende Systeme lernen, wie sie ihre Arbeitsweise verbessern oder ihre Datenbasis vergrößern. Solche Systeme verwenden beispielsweise bestimmte Induktionsheuristiken, die ein Programm in die Lage versetzen, Klassenbeschreibungen aus positiven und negativen Beispielen zu lernen (Winston 1987: 397-420).

Der Künstlichen Intelligenz Forschung unterliegen zwei grundsätzlich verschiedene Zielstellungen (vgl. z.B. Winston 1987: 21 oder Weizenbaum 1978: 219-223):

- die *ingenieurwissenschaftlich orientierte KI* will möglichst intelligente, praxisrelevante Problemlösungssysteme konstruieren, ohne sich darum zu kümmern, ob Menschen diese Aufgabe kognitiv genauso oder ähnlich ausführen. Das Ziel ist vereinfacht gesagt, "Computer nützlicher zu machen". Beispiele wären ein maschinelles Übersetzungssystem für Bedienungsanleitungen oder ein Expertensystem für Kraftfahrzeug-Reparaturen. Solche praktisch ausgerichteten Systeme erheben in der Regel nicht den Anspruch, die übertragenen Aufgaben nach dem Vorbild menschlicher kognitiver Prozesse zu lösen.
- die *modellierende KI* will menschliches Denken und Verhalten mit Hilfe von Computern und Programmen besser verstehen. Beispielsweise werden Modelle von Denkprozessen auf Maschinen nachgebildet, um aus diesen Modellen Rückschlüsse auf die Funktionsweise beim Menschen zu machen. Die modellierende KI verfolgt also nicht das Ziel, praktisch verwendbare, auf Leistung angelegte Programme zu erstellen, sondern Programme, die es ermöglichen, die Prinzipien menschlichen Denkens und Verhaltens zu verstehen. Frühe und bekannte Beispiele sind das in Kap. 3.1.1 vorgestellte Silbenlernprogramm EPAM von Feigenbaum (1962) oder das Paranoia-Modell von Colby (1975).

Wenn wir im folgenden von "KI-Modellen" sprechen, so sind immer Computerprogramme gemeint, in denen Realitätsausschnitte auf dem Rechner mit Werkzeugen, Methoden und Techniken der KI repräsentiert sind. Diese Modelle sind typischerweise in symbolisch orientierten KI-Sprachen wie LISP, PROLOG oder höheren Entwicklungswerkzeugen geschrieben und bedienen sich bestimmter KI-typischer Verfahren, wie Wissensrepräsentations- oder Planungstechniken. Wir verwenden den Term "KI-Modell" vorerst in dieser generellen Bedeutung. Später werden wir uns mit einer speziellen Form solcher KI-Modelle befassen, bei denen die Repräsentation von Wissen in Form von Regeln im Vordergrund steht.

Eine allgemeinere, weitergehende Behandlung der KI ist an dieser Stelle nicht möglich, da dies den Rahmen des Buches weit sprengen würde. Wir müssen uns auf die Nennung einiger Lehrbücher beschränken. Eher technisch orientierte KI-Einführungen sind Rich (1988), Winston (1987) und Tanimoto (1990). Einen allgemein einführenden Überblick gibt Waltz (1982). Eine Einführung in die (psychologisch orientierte) modellierende KI ist Boden (1987, 1988) oder Opwis (1992).

Eine grundlegende Bemerkung zum Verhältnis KI-Sozialwissenschaft sei jedoch noch angeführt. Unserer Ansicht nach bilden bestimmte Ideen der Künstlichen Intelligenz faszinierende Möglichkeiten und Aussichten für die Theorieentwicklung in den Sozialwissenschaften. Analog wie die kognitive Psychologie der KI entscheidende theoretische Impulse verdankt (und umgekehrt), ist es denkbar, daß die KI auf die Sozialwissenschaften ähnliche Implikationen ausübt (und umgekehrt). Erste Anfänge sind die in der soziologischen Literatur mehr und mehr rezipierten KI-Schema-Konzepte. Ein anderes Beispiel ist in jüngerer Zeit der Ansatz der "Verteilten KI", in dem Multi-Akteur-Systeme zur Entwicklung und Analyse intelligenter Gemeinschaften implementiert und untersucht werden (vgl. hierzu einführend von Martial (1992), Müller (1993) und Beiträge der KI 1/92). Verschiedentlich wird der Künstlichen Intelligenz für die Sozialwissenschaften eine ähnliche Rolle zugeschrieben, wie die Mathematik sie für die Naturwissenschaften hat. Beispielsweise prophezeit der Psychologe Allport: "Artificial intelligence will ultimately come to play the role vis-a-vis the psychological and social sciences that mathematics, from the seventeenth century on, has done for the physical sciences" (Allport 1980: 31). Die Informatiker Bibel und Siekmann verweisen darauf, daß die KI den Prozeß der wissenschaftlichen Entwicklung in analoger Weise unterstützen kann, wie es bestimmte formale Hilfsmittel im Hinblick auf die mathematische Modellierung bereits tun. Für die Kulturwissenschaften ist jedoch die KI-Modellierung "von erheblich größerer Bedeutung, als beispielsweise die Modellierung durch mathematische Gleichungen" (Bibel/Siekmann 1994: 22). Die potentiellen Auswirkungen der KI auf die theoretischen Sozialwissenschaften werden ausführlicher behandelt in Anderson (1989) und Manhart (1991).

3. Computerprogramme als Theorien

Bevor wir detaillierter auf KI-Modellierung und Wissensrepräsentation eingehen, befassen wir uns in diesem Kapitel mit einem wichtigen Aspekt computerunterstützter Theoriebildung, der eng mit der KI verknüpft ist. Gemeint ist die Auffassung, nach der Computerprogramme unmittelbar als Theorien angesehen werden. Diese Vorstellung ist besonders in der *modellierenden KI* verbreitet.

Grundsätzlich können bei der Computermodellierung zwei Vorgehensweisen unterschieden werden: entweder man geht von (empirischen oder fikten) Daten aus oder von einer bereits existierenden, umgangssprachlich oder mathematisch formulierten Theorie. Im letzten Fall würde das Computerprogramm die vorliegende Theorie "irgendwie" repräsentieren, und diese Wendung wurde auch bereits mehrmals gebraucht. Es soll nun die in der Literatur vielfach vertretene Auffassung untersucht werden, wonach das Programm direkt die Theorie *ist* oder die Theorie *verkörpert* - auch ohne ein verbales oder mathematisches Gegenstück. Diese Position findet sich vor allem in der KI-orientierten und manchmal in der nicht-KI-orientierten Simulationsliteratur. Wir möchten zunächst einige Autoren zitieren, welche diese Vorstellung mehr oder weniger explizit vertreten.

(Z1) Ostrom (1988) betrachtet in "Computer Simulation: The Third Symbol System" Computermodelle neben der natürlichen Sprache und der Mathematik als drittes Symbolsystem, in dem sich Theorien repräsentieren lassen. Ostrom behauptet, daß *jede* Theorie, die sich in einem der beiden anderen Systeme ausdrücken läßt, sich auch als Computerprogramm formalisieren läßt.

(Z2) Frijda (1967: 610) propagiert Computerprogramme als "unambigious formulations of a theory" und begründet dies mit der Präzision von Programmiersprachen.

(Z3) Die KI-Pioniere Simon und Newell (1956) argumentieren in einem frühen Artikel dafür, Theorien statt in verbaler oder mathematischer Form direkt als Computerprogramme zu formulieren.

(Z4) Weizenbaum (1978: 205-206) behauptet, daß "Modelle in Form von Computerprogrammen ebenfalls Theorien darstellen (zumindest verdienen einige Programme die Bezeichnung)".

(Z5) In einem der weitverbreitetsten KI-Lehrbücher findet man die folgende Bemerkung: "Occasionally after seeing what a program can do someone will ask for the theory behind it. Often the correct response is that the program *is* the theory" (Winston 1984: 12, Hervorhebung Winston).

(Z6) Ähnliche Anmerkungen finden sich gehäuft in Boden (1987), z.B. über ein psychologisches Neurosenmodell: "... the program represents a psychological theory of neurosis" (Boden 1987: 6).

(Z7) An anderer Stelle wird der dynamische Charakter von computerisierten Theorien betont: "A functioning program is a theory that is intended more as a movie of mind than a portrait of it..." (Boden 1987: 34).

(Z8) Newell und Simon sagen über das Potential einer Informationsverarbeitungs-Theorie menschlicher Problemlösung: "... the theory performs the tasks it explains. That is, a good information processing theory of a good human chess player can play good chess; a good theory of how humans create novels will create novels, a good theory of how children read will likewise read and understand" (Newell/Simon 1972: 10-11).

(Z9) Ähnlich äußert sich Wessels (1984: 371): Eine Simulation ist "in dem Ausmaß eine angemessene psychologische Theorie, indem die Simulation menschliches Verhalten exakt abbildet".

In der Regel werden diese Thesen ohne eine Begründung oder weitere Diskussion in den Raum gestellt. Explizit wird dieses Thema behandelt von Kobsa (1982) "On Regarding AI Programs as Theories", Kobsa (1984) "What is Explainded by AI-Models", Simon (1979) "Philosophical Objections to Programs as Theories" und Manhart (1989) "Können AI-Programme als Theorien betrachtet werden?".

Wenn Programme mit Theorien gleichgesetzt werden (können), drängen sich einige Fragen auf, z.B.:

1. Läßt sich, wie Ostrom ohne weitere Begründung behauptet, *jede* Theorie in ein Programm übersetzen (Z1)?
2. Wenn ja, ist das *ganze* Programm (Z5) oder sind nur Programmteile mit der Theorie *identisch*?
3. Sind Programmiersprachen überhaupt *geeignete Sprachmittel* zur Repräsentation von Theorien?
4. Was ist die Beziehung zwischen verbaler/mathematischer Theorie, Computermodell und Computertheorie (z.B. Z4)?
5. Ist I/O-Äquivalenz von empirischem System und Programm hinreichend für den theoretischen Status eines Programms (Z8, Z9)?

3.1 Von der Theorie zum Programm: theoriegesteuertes Vorgehen

Wir unterscheiden im folgenden zwei grundsätzlich verschiedene Vorgehensweisen, die man als Top-Down- und Bottom-up-Ansatz bezeichnen kann. Im Top-Down-Ansatz geht man von einer vorliegenden Theorie aus und versucht, diese in ein lauffähiges Computerprogramm zu bringen. Beim Bottom-up-Ansatz schreibt man zunächst experimentelle Programme, die empirische Phänomene beschreiben, verfeinert diese, bis das Programm "funktioniert" und kann unter Umständen das Ergebnis - so die Behauptung von z.B. Z7, Z9 - als Theorie der abgebildeten Phänomene auffassen. Diese Dichotomie korrespondiert in etwa der Unterscheidung von Langley et al (1987) in "Data-Driven and Theory-Driven Science". Das Ziel beider Vorgehensweisen ist die wissenschaftliche Entdeckung: bei der datengesteuerten Methode die Entdeckung einer Menge von Regularitäten oder einer Theorie (Langley et al 1987: 23), beim theoriegesteuerten Vorgehen neue, überraschende Ableitungen aus der Theorie.

3.1.1 Maschinelle Theoriepräsentation

Die allgemein akzeptierte Turing-These behauptet, daß man jedes intuitiv effektive Verfahren mit einer Turing-Maschine realisieren kann und umgekehrt, daß alles, was auf der Turing-Maschine ausführbar ist, algorithmisch ist. Da im Prinzip jeder moderne Computer eine Turing-Maschine ist (Weizenbaum 1978: 93), folgt daraus, daß man eine Theorie dann auf einer Maschine ausführen kann, wenn es gelingt, diese in effektiver Form, also algorithmisch, zu formalisieren. Lindenberg (1971: 85) deutet die Turing-These salopp in der Form, daß uns der Computer zu Hilfe kommt, wenn wir statt in Propositionen in Algorithmen denken. Die Antwort auf die Frage, ob eine Theorie in ein Computerprogramm übersetzbar ist, wäre also: ja, wenn es gelingt, einen Algorithmus für eine Theorie zu finden (die Theorie zu "algorithmisieren") - wobei wir weiter unten diese Behauptung etwas differenzieren. Das Problem, das sich stellt, ist, ob es zu einer gegebenen - mathematischen oder verbalen - empirischen Theorie eine algorithmische Form *geben kann*. Da Theorien im Sinn des Standardtheorienkonzepts als Satzsysteme mit nomologischen Aussagen betrachtet werden, reduziert sich die Frage im wesentlichen darauf, ob Gesetze in algorithmischer Form dargestellt werden können und Programmteile entsprechend diese Gesetze repräsentieren.

Anhand der Ausführungen in Kap. 2.2 wurde deutlich, daß es für bestimmte Probleme keinen Algorithmus geben kann. Somit kann es auch Theorien geben, für die es nicht möglich ist, eine algorithmische Form anzugeben. Eine algorithmische Theorie, die irgendwelche Aussagen über natürliche Zahlen entgegennimmt und mit wahr oder falsch antwortet, kann nach dem Gödel-Theorem nicht existieren. In den Naturwissenschaften und in der Mathematik dürfte es viele Beispiele für Theorien geben, die nicht in Programme transformiert werden können. Die Aussage von Ostrom (Z1), nach der *jede* Theorie in ein Programm übersetzt werden kann, ist damit eindeutig falsch.

Beschränkt man sich auf human- und sozialwissenschaftliche Theorien, dürften die Theoreme der Berechenbarkeitstheorie weniger relevant sein. Statt dessen werden die zu bearbeitenden Probleme oft nicht verstanden und haben intuitiven Charakter. Ist das Verständnis jedoch hinreichend, müßte die Übersetzung gelingen. Die folgenden Projekte zeigen, daß derart in eine Programmiersprache transformierte Theorien *unmittelbar* als programmiersprachliche Darstellung der Theorie betrachtet werden.

Ein frühes Beispiel ist EPAM (Elementary Perceiver and Memorizer) von Feigenbaum (1963). EPAM modelliert die kognitiven Prozesse, die beim Lernen von Unsinnssilben vor sich gehen. Das Modell geht davon aus, daß das Lernen von Unsinnssilben ein komplexer Prozeß der Symbolmanipulation ist, der in Form noch elementarerer Symbolmanipulationen beschrieben und verstanden werden kann. Das Modell erklärt z.B., wie es möglich ist, daß wir längere Zeit etwas völlig vergessen und uns trotzdem später wieder daran erinnern können (Weizenbaum 1978: 218). Diese Erklärung erfolgt mit Hilfe nomologisch-theoretischer - z.T. statistisch formulierter - Prinzipien etwa folgender Art (Feigenbaum 1963: 299):

"Je ähnlicher Reizsilben einander sind, um so mehr Versuche sind zum Erlernen erforderlich" oder
"Wenn Assoziationen über eine Anzahl von Versuchen hinweg korrekt wiedergegeben werden, dann werden sie manchmal vergessen, um wieder aufzutreten und erneut zu verschwinden".

Diese Prinzipien oder Gesetzmäßigkeiten finden sich im Programmcode und das Programm kann damit als eine Theorie darüber betrachtet werden, wie Menschen Nonsens-Silben lernen: "Wenn man es beispielsweise einem Psychologen gibt, der mit der Programmiersprache vertraut ist, in der sie geschrieben ist, so darf man erwarten, daß er sie verstehen wird. Was es jedoch zur Theorie macht, ist der Umstand, daß es bestimmte Prinzipien aufstellt, aus denen Konsequenzen gezogen werden können. Diese Prinzipien liegen in Form eines Computerprogramms vor, und ihre Konsequenzen lassen sich am Verhalten des Programms ablesen, d.h. an der Art und Weise, wie der Computer das Programm liest" (Weizenbaum 1978: 235-236).

Was das Programm "zur Theorie macht", ist also das Vorliegen bestimmter Prinzipien oder *nomologischer Aussagen in Form des Programmcodes*. Colby (1973) bezieht sich in einem Modell, das einen paranoiden Akteur simuliert, direkt auf das nomologische Erklärungsschema der analytischen Wissenschaftstheorie:

"This model is considered a theoretical model because its inner structure embodies an explanatory account of the complex phenomena of paranoid communicative behavior. An explanatory account contains statements of lawlike generalisations. In order to explain concrete individual events, it also contains singular statements of individual initial conditions. In order to run and test the model, the general lawlike principles must be combined with the singular conditions to generate the I-O behavior of this individual hypothetical case" (Colby 1973: 265-266).

Die von Colby postulierten Parallelen zwischen dem D-N-Erklärungsschema und dem Input-Output-Schema der Computersimulation lassen sich genauer wie folgt darstellen:

Theorie (D-N-Schema)	*Computermodell (I-O-Schema)*
Randbedingung	Input
Gesetz	Programm
----------L	==========A
Ereignis	Output

Den Randbedingungen entsprechen die Inputdaten, den Gesetzen das Computerprogramm und den aus Randbedingungen und Gesetzen folgenden Ereignissen die Outputdaten des Computers. Die zwei wesentlichen Aufgaben des D-N-Schemas sind Erklärung und Prognose. Im D-N-Schema gilt ein Ereignis dann als *erklärt*, wenn es aus gesuchten Anfangsbedingungen und Gesetzen logisch deduziert werden kann. Dem entspricht auf Computerseite die "Erklärung" des Outputs durch "Herleitung" aus dem Programm und den Inputdaten. Umgekehrt - und für Computersimulation wichtiger - können im D-N-Schema Ereignisse aus gegebenen Randbedingungen und

Gesetzen *prognostiziert* werden. Dies korrespondiert beim Computermodell der Erzeugung von Outputdaten aus Inputdaten und dem Programm, so daß das Computerprogramm als Vorhersageinstrument verwendet werden kann.

Dennoch sind D-N- und I-O-Schema gewöhnlich nicht völlig äquivalent. Während im D-N-Schema der Übergang vom Explanans zum Explanandum auf einem logischen Schluß basiert (symbolisiert durch ein L an der Übergangslinie), unterliegt dieser Überführung beim Computermodell normalerweise keine logische Deduktion, sondern sie *ergibt sich aus dem Algorithmus des Programms* (symbolisiert durch ein A an der Übergangslinie). Das einschränkende "normalerweise" ist deshalb nötig, weil es logische Programmiersprachen gibt, bei denen der Output unmittelbar als logische Deduktion aufgefaßt werden kann (vgl. hierzu Kap. 4.2.2). Die Erzeugung des zu erklärenden oder prognostizierenden Ereignisses/Outputs findet aber bei beiden Schemata gewöhnlich auf unterschiedliche Weise statt. Diese Divergenz sollte man nicht aus den Augen verlieren, wenn man Computerprogramme als Theorien betrachtet.

Im metatheoretischen Rahmen des hier unterlegten Standardtheorienkonzepts macht die Aussage, daß ein Programm eine Theorie ist oder verkörpert, also Sinn, und wir verwenden für derartige Programme im folgenden auch die Bezeichnung "Computational Theories".

Weizenbaum (1978: 196ff.) sieht in der computerisierten Darstellung von Theorien eine völlig neue Beziehung zwischen Theorie und Modell: Theorien in Form von Computerprogrammen sind unter dem Gesichtspunkt der Sprache - abgesehen von einigen Besonderheiten - völlig gewöhnliche Theorien. Ein Physiker kann z.B. seine Theorie über das Pendel entweder als Menge mathematischer Gleichungen oder als Computerprogramm formulieren. Das Programm hat aber den Vorteil, daß es nicht nur von jedem verstanden werden kann, der in der Sprache ausgebildet ist, sondern darüber hinaus, daß es auch einen Computer durchlaufen kann. Da ein Modell den Verhaltensgesetzen einer Theorie genügt, *ist eine Theorie in Form eines Programms damit sowohl eine Theorie als auch - in einen Computer eingegeben und von diesem bearbeitet - ein Modell, auf das die Theorie Anwendung findet.* In diesem Sinn interpretiert Weizenbaum die Aussage von Newell und Simon über ein Programm, das menschliches Problemlösungsverhalten nachbildet: "Die Theorie löst die Aufgaben, die sie erklärt" (Z8). Eine Theorie kann nichts lösen, wohl aber ein Modell. In Computational Theories wird die Theorie - in der Interpretation von Weizenbaum - damit zum Modell und umgekehrt, so daß Theorie und Modell identisch werden. Wir werden später in Kap. 5.4 sehen, daß eine Gleichsetzung von Theorie und Modell aus der Sicht der strukturalistischen Wissenschaftstheorie jedoch nicht sinnvoll ist.

3.1.2 Probleme maschineller Theorierepräsentation

Mit der Tatsache, daß die Theorie an die Bedingung der Ausführbarkeit auf Computern angepaßt werden muß, sind einige pragmatische Besonderheiten und Probleme verknüpft.

Lindenberg (1971) weist darauf hin, daß in der Praxis Theorien selten unter dem Gesichtspunkt der Äquivalenz übersetzt werden.

- Erstens arbeitet man meist nicht mit der ganzen Originaltheorie, sondern begnügt sich nur mit bestimmten Ableitungen, deren Resultat für Testkomponenten anderer Sätze gebraucht wird. Auf diese Weise werden Ableitungen einzelner Sätze zu einem Prozeßablauf zusammengefügt und die (Teil-) Theorie wird "prozessionalisiert".
- Beispiele zeigen, daß zweitens oft zusätzliche Annahmen in das Computerprogramm eingeführt werden, die in der Originaltheorie nicht enthalten sind.

Damit haben wir die Situation, daß in der programmierten Theorie Elemente der Originaltheorie nicht enthalten sind und Elemente in Form von Zusatzannahmen hinzukommen. Dies läßt sich in folgendem Mengendiagramm veranschaulichen.

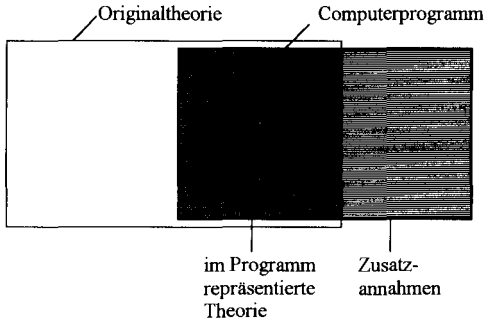


Abb. 3.1: Beziehung zwischen Originaltheorie und (in ein Computerprogramm) übersetzter Theorie

Die Umsetzung einer Theorie in ein Programm und insbesondere das Hinzufügen von Zusatzannahmen sollen an einem konkreten Beispiel verdeutlicht werden.

Homans (1961) stellt in "Social Behavior: Its Elementary Forms" eine Theorie sozialen Verhaltens vor, welche Prinzipien der klassischen Ökonomie und behavioristischen Psychologie vereint. Soziales Handeln wird auf der Grundlage von Lerntheorie und Nutzenaustausch erklärt. Der Theorie und dem Modell unterliegen zwei zentrale Annahmen:

(1) Belohntes Verhalten wird häufiger gezeigt als nicht belohntes;
(2) Jede soziale Interaktion ist durch einen Austausch von Belohnungen gekennzeichnet.

Homans formuliert seine Theorie in fünf Hypothesen, und Gullahorn/Gullahorn (1963) haben die Propositionen aus Homans (1961) in ein Modell (HOMUNCULUS) übersetzt und - in der Terminologie Lindenbergs - "prozessionalisiert".

Das Modell realisiert die Theorie mit einer einfachen Interaktionsfolge zwischen zwei Akteuren, die gemäß den Homansschen Thesen handeln. Ein Akteur wird dabei als hypothesentestender und informationsverarbeitender Organismus betrachtet, der Informationen empfangen, analysieren, rekonstruieren und speichern kann. Der Ablauf der Simulation gibt eine Folge von Interaktionen wieder, in denen ein Akteur A einen Akteur B um Hilfe bittet, Hilfe erhält, dafür Beifall und Respekt als Belohnung zollt,

was die Interaktion verstärkt, bis der Helfende sie eventuell abbricht, weil sie ihn mehr kostet, als sie ihm an Belohnung einbringt.

Lindenberg (1971) zeigt, wie in das Modell Annahmen verschiedener Art eingeführt werden, die in der Originaltheorie nicht enthalten sind. Diese Annahmen, die wir im folgenden kurz betrachten wollen, sind bestimmte Randbedingungen, Operationalisierungen und Spezifizierungen.

Da Homans sich an der behavioristischen Lerntheorie orientiert, wurden *alle kognitiven Aspekte in die Randbedingungen geschoben.* Im Modell werden diese Aspekte aber explizit gemacht. Homans erste These lautet:

"Wenn das Verhalten in einer früheren Stimulussituation belohnt worden ist, dann wird diese Person jetzt dies oder ein ähnliches Verhalten um so wahrscheinlicher an den Tag legen, je mehr die frühere Stimulussituation der jetzigen ähnelt" (Homans 1961: 53).

Gullahorn/Gullahorn (1963) fügen nun die Hypothese hinzu, daß die Person sowohl Stimuli als auch Reaktionen *generalisieren kann.* Diese Annahme ist nicht explizit in der These enthalten, sondern nur implizit ("dies oder ein ähnliches Verhalten", "je mehr die frühere Stimulussituation der jetzigen ähnelt"). In das Modell wird damit eine Gedächtnisstruktur eingeführt, die in der Originaltheorie nicht enthalten ist.

Ein zweiter Typ von Annahmen, die Gullahorn/Gullahorn machen, betrifft *Operationalisierungen.* Homans zweite These lautet:

"Je häufiger innerhalb einer gegebenen Zeitperiode jemand durch eine Handlung das Handeln eines anderen belohnt hat, um so öfter wird der andere die Handlung ausführen" (Homans 1961: 54).

Eine einfache Operationalisierung bestünde im Zählen der belohnten Reaktionen, Gullahorn/Gullahorn lehnen dies aber als der menschlichen Informationsverarbeitung inadäquat ab. Sie postulieren einen viel gröberen Maßstab in Form einer Ordinalskala mit fünf Punkten von "fast immer belohnt" zu "fast nie belohnt". Im Modell wird also eine *Ad-hoc-Annahme* verwendet, die zwar plausibel erscheint, aber in der Theorie nicht vorkommt.

Die dritte Art von Annahme, die die Modellbauer einführen, ist *Spezifizierung*, was Lindenberg an der dritten Hypothese erläutert:

"Je mehr Nutzen jemand durch eine Handlung eines andern hat, um so häufiger wird er so handeln, daß er von anderen mit dieser Handlung belohnt wird" (Homans 1961: 55).

Im Modell wird spezifiziert, *was* einer Person Nutzen bringt und *welche Verhaltenseinheiten* für ihn mehr Nutzen bringen als andere. Es werden Parameter geschaffen, die bei der Simulation mit eindeutigen Werten besetzt werden.

Lindenberg (1971) diskutiert die Beispiele so, wie wenn die Einführung dieser Annahmen eine *zwangsläufige* Folge der Computerimplementierung wäre. Die Einschleusung der Zusatzannahmen wird durchweg in dem Sinn gedeutet, daß *jede* in ein Computerprogramm prozessionalisierte Theorie viel mehr enthalten muß als das verbal oder mathematisch formulierte Gegenstück und in der Regel erheblich komplizierter wird als das Original. Dies ist jedoch nur bei einem bestimmten Implemen-

tierungstyp der Fall, den Gullahorn/Gullahorn gewählt haben und bei dem versucht wird, die Theorie möglichst weit in empirischen Daten zu verankern.

Unserer Ansicht nach ergibt sich aus den Beispielen eine ganz andere Folgerung. Sie belegen, daß sehr abstrakt formulierte Theorien um so schwieriger und aufwendiger zu implementieren sind, je mehr man versucht, sie empirisch zu verankern. Wenn man hingegen "theoretische" Daten zuläßt, gibt es die angesprochenen Probleme nicht, d.h. es müssen weder Randbedingungen, Operationalisierungen noch Spezifizierungen eingeführt werden. Läßt man als theoretisches Datum z.B. zu:
Person x hat von Handlung y der Person z einen Nutzen u (formal: u(x,y,z)),
dann braucht nicht spezifiziert zu werden, was einer Person Nutzen bringt und Daten des Typs
u(x,y,z1) > u(x,y,z2)
können direkt im Bedingungsteil der obengenannten Hypothese verwendet werden. Je mehr man aber die Theorie empirisch verankern will, um so mehr Aufwand ist bei der Umsetzung in ein Programm nötig und um so mehr Zusatzannahmen müssen eingeführt werden.

Die Konsequenz dieses praktischen Beispiels ist, daß bei der Umsetzung einer Theorie in ein Programm verschiedene Implementierungsebenen unterscheidbar sind: je abstrakter und höher die "theoretische Ebene", um so weniger werden die von Lindenberg angesprochenen Probleme relevant. Je konkreter und "empirischer" aber die Ebene, um so mehr treten diese Schwierigkeiten auf.

Ein zweites Problem maschineller Theoriepräsentation ist die unterschiedliche Wissensorganisation von Theorien und Programmen. Konventionelle Computerprogramme haben grundsätzlich einen *anderen* formalen Wissensaufbau als (qualitative) Theorien. Der hierarchische Aufbau von Computerprogrammen (vgl. Kap. 2.3, S.33-34) erzwingt, daß eine qualitative Theorie in eine algorithmische, abgestuft strukturierte Sequenz von Befehlen und Abfragen übersetzt werden muß. Der Programmierer muß ferner einen schrittweisen Ablauf von vorzunehmenden Operationen festlegen (vgl. Puppe 1988: 3, Schnupp/Nguyen Huu 1987: 17). Dies steht in krassem Gegensatz zum Standardtheorienkonzept, nach dem eine Theorie im wesentlichen eine Menge von nomologischen Aussagen oder Propositionen ist und statisches, deklaratives Wissen enthält. Während Algorithmen also prozedurales, algorithmisches Wissen enthalten, die spezifizieren, *wie etwas* in welcher Reihenfolge *getan* wird, enthalten Theorien propositionales Wissen, die Sachverhalte in Aussagesätzen codieren und ausdrücken, *was gelten* soll (vgl. die Homans-Hypothesen). Wenn Theorien in Algorithmen überführt werden müssen, wird der Modellierer damit gezwungen, das "Was" durch ein "Wie", die Proposition durch eine Prozedur auszudrücken.

In konventionellen Programmen kann diese "semantische Lücke" zwischen der verbal oder mathematisch formulierten Theorie und dem Computerprogramm eine erhebliche Behinderung sein. Der Ausdruck "semantische Lücke" bezieht sich dabei auf den Unterschied zwischen der ursprünglichen Repräsentation der Theorie und der programmierten Version (vgl. Merritt 1990: 3-4). Die semantische Lücke zwischen einem quantitativen Gesetz und der Repräsentation in einer numerischen Sprache wie

FORTRAN ist gering. Die Differentialgleichung von Abb. 2.1 (S.18) kann z.B. unmittelbar ohne wesentliche Änderung in die numerische Sprache abgebildet werden. Hingegen ist die semantische Lücke zwischen den Propositionen von Homans und einer konventionellen Programmiersprache erheblich.

Ein drittes Problem von Computational Theories ist die Trennung theoretisch relevanter von theoretisch nicht relevanten Prozeduren. Während eine Theorie, die in einer Programmiersprache ausgedrückt wird, theoretische Prozeduren enthalten muß, ist eine Vielzahl von Prozeduren eines Programms *irrelevant* für die Theorie. Da die Theorie auf der Maschine ausgeführt wird, werden Prozeduren für "Buchführungszwecke" (Lindenberg 1971: 92) benötigt, auf die - bedingt durch die maschinelle Ausführung - nicht verzichtet werden kann. In realen Programmen werden ganze Teilprozeduren für Zwecke gebraucht, die in keiner Relation zur Theorie stehen: "The low order subroutines and a number of technical necessities are determined by the particularities of the programming language, the mode of operation of the particular computer, and the special limitations inherent in serially operating digital machines" (Frijda 1967: 611). Weiter muß es Prozeduren geben, die die Kommunikation mit der Außenwelt regeln, die Benutzerschnittstelle definieren oder Daten einlesen. Es ist klar, daß diese Module nichts mit einer Theorie zu tun haben. Faßt man das ganze Computerprogramm als Modell auf, so ist die Abundanzmenge, anders als bei Nicht-Computermodellen, hier besonders groß. Da es theoretisch relevante und nicht relevante Programmteile gibt - die unter Umständen sogar vermischt werden - macht es wenig Sinn, die ganze Summe von Prozeduren als Theorie anzusehen und das Programm mit der Theorie gleichzusetzen. Das Programm *ist* also nicht die Theorie (vgl. Z5), vielmehr sollte das Programm besser als *Repräsentation* einer Theorie betrachtet werden (Frijda 1967: 611).

Ein Programm, das eine Theorie repräsentiert, enthält damit Teile der Originaltheorie plus eine (eventuell leere) Menge von Zusatzannahmen plus Teile, die nichts mit der Theorie zu tun haben. Das oben angegebene Mengendiagramm kann danach wie in Abb. 3.2 erweitert werden.

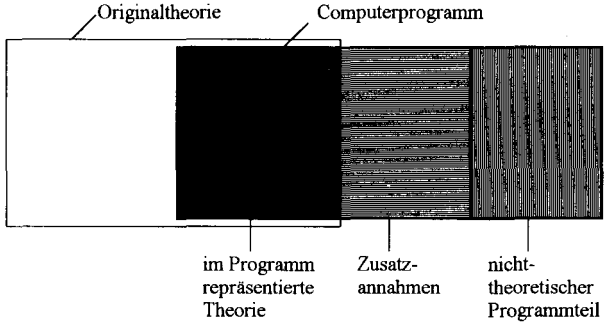


Abb. 3.2: Die Beziehung zwischen Originaltheorie und als Computerprogramm repräsentierter Theorie unter Berücksichtigung nicht-theoretischer Programmteile

Frijda (1967) fordert die strikte Trennung theorierelevanter von nicht theorierelevanten Prozeduren: "The theory-relevant routines must, so to speak, be isolated from theoretically irrelevant auxiliary operations and must be as independent of technical realizations as possible" (Frida 1967: 612). Unglücklicherweise ist die Abschottung theorierelevanter Prozeduren von theoretisch irrelevanten Bearbeitungsprozeduren in konventionellen Sprachen nicht streng durchzuführen, da *keine* "klare Schnittstelle" zwischen den theoretischen Programmteilen und den Problemlösungsstrategien, die die theoretischen Teile bearbeiten, existiert (vgl. Puppe 1988: 2-3).

Ungeachtet dieses Problems, für das weiter unten eine Lösung angeboten wird, ist damit vieles im Programm keine Theorie und nichts deutet an, wann etwas zur Theorie gehört und wann nicht (Frijda 1967). Es liegt somit am Programmautor klar zu machen, welche Programmteile die Theorie verkörpern und wie und unter welchen Bedingungen diese funktionieren.

Damit kommen wir zu einer vierten und letzten Streitfrage bei Computational Theories: dem Kommunikationsproblem. Die meisten zeitgenössischen Autoren stehen auf dem Standpunkt, daß es grundsätzlich sinnvoll und nützlich ist, das Computerprogramm in der Wissenschaftlergemeinschaft zu kommunizieren und nicht nur die vom Modell produzierten Ergebnisse. Die Frage ist, in welcher Form dies geschehen soll.

Die bloße informelle Programmbeschreibung war die in der Vergangenheit bevorzugte Kommunikationsform. In der Simulationsliteratur der sechziger und siebziger Jahre findet sich kaum Programmcode. Gullahorn/Gullahorn (1963 und 1965) z.B. stellen ihr Programm informell vor anhand von Struktogrammen und verbalen Beschreibungen. Diese Präsentation unterscheidet sich in nichts von verbalen Theorien und der Vorteil der Präzisierung geht dadurch eindeutig verloren.

In der Vergangenheit gab es wichtige Gründe, auf die Publizierung von Programmtext zu verzichten: teure, inkompatible Maschinen an den Universitäten; nicht standardisierte und hardwarenahe Programmiersprachen; schwer lesbare, lange Programme. Viele Gründe für die Nichtveröffentlichung von Programmcode haben sich historisch überholt durch die Entwicklung von Hardware-Standards, strukturierter und normierter Programmiersprachen sowie nicht zuletzt veränderter Wissenschaftler-Sozialisation. Mehr und mehr Fachwissenschaftler beherrschen zumindest in Grundzügen eine Programmiersprache, Programme sind einfacher und leichter lesbar geworden, und Sprachwerkzeuge und Hardware sind allgemein verfügbar, standardisiert und billig.

Die Veröffentlichung des Programmtextes hat zwei entscheidende Vorteile:

- Der Programmcode kann gelesen werden und die Übersetzung der Theorie kann geprüft werden.
- Andere Wissenschaftler verwenden und modifizieren das Programm für Simulationen und Experimente, was die Weiterentwicklung der Theorien fördert.

Unter dem Aspekt der Lesbarkeit und Modifizierbarkeit sollten die Programme aber neben der Trennung theoretisch relevanter von theoretisch nicht relevanten Prozeduren möglichst *gut dokumentiert* und *informell beschrieben* sein. Ein bleibender

Nachteil ist sicher die oft erhebliche Länge der Programmtexte von mehreren 1.000, 10.000 oder gar 100.000 Zeilen. Dies läßt sich durch Veröffentlichung auf Datenträger (Disketten, CDs) umgehen, was den Vorteil hat, daß der Code gleich in maschinenlesbarer Form vorliegt. Eine wichtige Vermittlerrolle für Computermodelle dürften zukünftig vor allem wissenschaftliche Computernetze wie Internet spielen.

3.2 Vom Programm zur Theorie: datengesteuertes Vorgehen

Wir sind bislang davon ausgegangen, daß eine *bereits vorliegende* - umgangssprachlich oder mathematisch formulierte - Theorie in ein Computerprogramm übersetzt wird.

Aus der Literatur läßt sich nun ein zweiter Ansatz rekonstruieren, der genau umgekehrt vorgeht: ausgehend von den Daten bzw. dem I/O-Verhalten des betrachteten empirischen Systems werden Programme gebaut, die dieses möglichst gut nachahmen. Der Modellierer beschreibt sein Modell dabei idealerweise direkt in der Sprache des Simulationssystems (Möhring 1990: 23), d.h. Modelle werden ohne - explizit - vorliegende Theorie gleich im Rahmen von Generate-and-Test-Programmen erstellt. Mit "Generate-and-Test" ist gemeint, daß Programmcode geschrieben wird, der versucht, das I/O-Verhalten des zu modellierenden Systems - zunächst grob - zu generieren. Mit dem generierten Code wird das I/O-Verhalten getestet und solange verbessert, bis das Verhalten dem empirischen System entspricht. Dieses Verfahren korrespondiert in etwa dem "rapid prototyping" der Informatik: der Entwurf einer Lösung und ihr rasches Umsetzen in ein Programm (Lischka/Diederich 1987: 28). Für bestimmte Problemstellungen wird also nicht explizit ein verbales oder mathematisches Modell formuliert, sondern diese werden gleich im Rahmen von Simulationsprogrammen angegangen. Das Ziel ist, durch das Erstellen der Programme theoretische Einsichten in die Funktionsweise des modellierten Gegenstands zu bekommen oder Gesetzmäßigkeiten zu entdecken. In Anlehnung an Ziegler (1972: 285) könnten wir dieses Vorgehen als den "Entwurf einer Theorie im Prozeß der Formalisierung" bezeichnen.[23]

Langley et al (1987) beschreiben dieses datengesteuerte Vorgehen in ihrem Buch "Scientific Discovery" anhand von Problemlöseprozessen wie folgt:

"We can write in an information processing language - a symbolic rather than numerical language - a computer program that describes, say, the processes that intelligent adults are hypothesized to use in solving some class of problems. We can then present the program and some human subjects with identical problems and compare their behaviors. From the computer we will obtain a trace, from the human subjects a verbal and written protocol of their behavior while they were solving the problems. We can then test the program - our theory of the behavior - by comparing

[23] Die von uns vorgenommene Trennung - von der Theorie zum Programm und umgekehrt - ist natürlich etwas künstlich und findet in diesem Idealtypus gewöhnlich nicht statt. Beispielsweise können in Generate-and-Test-Modelle bestimmte Aspekte schon vorliegender Theorien integriert werden und umgekehrt können in Programme, welche Theorien repräsentieren, eigene Ideen des Modellierers einfließen.

the trace with the protocol, just as any theory is tested by comparing the system path it predicts with data showing the actual path" (Langley et al 1987: 33).

Dieses Vorgehen ist typisch in der KI-basierten Modellierung. Ein Beispiel für ein solches Programm mit sozialwissenschaftlichem Bezug ist TEAMWORK (Doran 1985).[24] Das Modell untersucht, wie kooperative Aktivitäten von mehreren Akteuren koordiniert werden und simuliert hierzu ein Multi-Akteur-System, dessen Individuen eine gemeinsame Aufgabe bearbeiten müssen. Modelliert werden Planung und Ausführung von Aufgaben, sowie die Mechanismen von Kommunikation und Kooperation zwischen Akteuren. Explizites Ziel dieses Modells ist, ein tieferes Verständnis von realen Systemen zu bekommen "... both those that might be constructed and those human systems that are in existence around us" (Doran 1985: 160). Diese Fragestellungen würden Substanzwissenschaftler normalerweise durch Beobachtung und Experiment mit "natürlichen" Akteuren untersuchen. In der KI wird einfach ein Computerprogramm gebaut, das die Ideen der Modellierer ausdrückt, und es wird geprüft, ob sich das Programm wie erwartet verhält (Gilbert/Heath 1985: 1). Tut es dies nicht, wird das Modell solange verbessert, bis es den Erwartungen der Modellierer genügt.

Programme, die erfolgreich arbeiten, werden dabei oftmals als Theorien und Erklärungen der modellierten Phänomene betrachtet, wie die Zitate Z5, Z7 und Z9 und die Aussage von Langley et al. ("das Programm ist unsere Verhaltenstheorie") belegen. Wir wollen uns nun mit dieser Sichtweise befassen, die insbesondere typisch für die Cognitive Science ist, einer Disziplin, die im wesentlichen ein Zusammenschluß von KI und kognitiver Psychologie ist.

3.2.1 Computational Theories in der Cognitive Science

Cognitive Science (Kognitionswissenschaft) ist ein junges interdisziplinäres Gebiet, dessen Ziel die Erklärung technischer *und* menschlicher intelligenter Systeme ist (Simon 1981)[25]. Die Wurzeln dieser Disziplin ruhen in der Erkenntnis, daß sich Psychologen, Linguisten, Informatiker und Neurologen ähnliche Fragen über den menschlichen Geist aus unterschiedlicher Sichtweise stellen. In der Regel versteht sich die Cognitive Science als theoretische Psychologie, in der der Computer als Instrument dient, theoretische Vorstellungen zu realisieren und das Verhalten dieser Realisationen wiederum als Phänomen zu untersuchen. Das Erstellen von Computermodellen ist dabei nicht Forschungsziel, sondern Forschungsmethode.

Vorrangiger Untersuchungsgegenstand der Cognitive Science sind menschliche Mentalstrukturen, die auf einer mentalen Ebene erklärt werden ohne Rückgriff auf neuronale Zustände oder beobachtbares Verhalten. Bei der Analyse mentaler Prozesse bedient sich die Cognitive Science Konzepten der Computerwissenschaft: der Mensch wird in Analogie zum Computer als Informationsverarbeitungssystem verstanden, der Informationen aufnimmt, verarbeitet und abgibt. Die "Computermetapher" besagt, daß

[24] Eine zusammenfassende Beschreibung des Modells findet sich in Manhart (1991).

[25] Die Cognitive Science ist an etwa 20 amerikanischen und britischen Universitäten als eigenes Fach institutionell repräsentiert (Kobsa 1986: 116).

der Computer als Vorstellungshilfe bei der Beschreibung psychologischer Phänomene dienen kann, da die elementaren Prozesse, die der menschlichen Informationsverabeitung zugrunde liegen, denen bei Computern entsprechen. Mensch und Computer werden als Instanzen einer abstrakten Gattung symbolverarbeitender Systeme ("physikalischer Symbolsysteme") angesehen, in denen sich dieselben theoretischen Strukturen realisieren. Postuliert wird eine fundamentale Gemeinsamkeit von Berechnung und Denken: "Wenn wir ein Buch lesen, nehmen wir visuelle Informationen - die gedruckten Buchstaben - auf und encodieren sie nach ihrer Bedeutung. Wir erinnern uns an viele Informationen, die wir aufgenommen haben, und können selbst über die Zeit hinweg gespeicherte Informationen betrachten. Beim Erinnern rufen wir vorher gelernte Informationen ab, und Vergessen kann als fehlerhaftes Abrufen aufgefaßt werden. Man kann Menschen wie auch Computer als Systeme verstehen, die symbolische Informationen verarbeiten" (Wessels 1984: 39).

Die Cognitive Sciene geht aber weiter, als den Computer lediglich als Metapher für menschliche kognitive Prozesse zu benutzen. Ihr unterliegt die starke Annahme, daß *jedes psychologische Phänomen durch ein effektives Verfahren erzeugbar* (Boden 1988: 5)[26] und damit auf einem Computer ausführbar ist. Die Annahme ist insofern stark, als damit behauptet wird, daß alle psychischen Vorgänge zur Klasse der berechenbaren und - noch stärker - sogar zur Klasse der berechen- *und* durchführbaren Probleme gehören (vgl. Kap. 2.2 und Abb. 2.7, S.28). Das Ziel der Cognitive Science ist es, diese Algorithmen zu identifizieren und zu studieren. Sie versucht deshalb, Computerprogramme in Form von symbolverarbeitenden Algorithmen zu erstellen, die ein analoges Input-Output-Verhalten generieren wie die zu modellierenden menschlichen Phänomene. Um Einblick in die Funktionsweise menschlichen Denkens zu bekommen, wird versucht, dessen Funktionalität künstlich herzustellen. Kann menschliches Intelligenzverhalten Stück für Stück so simuliert werden, daß simuliertes und simulierendes Verhalten ununterscheidbar sind, so ist ein Beweis geglückt, daß Ergebnisse dieser Klasse von einem Computer berechnet werden können (Boden 1988: 7). Die Generierungsstrukturen der Modelle werden dabei nicht mehr als psychologisch irrelevant betrachtet, vielmehr läßt sich aus abstrakten Software-Strukturen von erfolgreichen Programmen unter Umständen auf Mental- und Verhaltensstrukturen bei Menschen schließen (Kobsa 1986: 113-114). In dieser Interpretation unterscheidet sich die Cognitive Science nicht grundlegend von dem Modellierungsziel, wie es in anderen empirischen Wissenschaften formuliert ist.

In ihrer radikalsten Variante geht die Cognitive Science aber noch weiter. Die Behauptung der "starken Variante" der Cognitive Science ist, daß die Programme *nicht nur ein wirksames Werkzeug* bei der Erforschung des menschlichen Geistes sind, sondern die menschlichen kognitiven Fähigkeiten *vollständig erklären*. Der Erklärungsbegriff der Cognitive Science unterscheidet sich dabei *fundamental* vom

---

[26] Boden (1988) spricht nicht von Cognitive Science, sondern von Computational Psychology. Gemeint ist aber im wesentlichen dasselbe. In dem Buch von Boden sind Computational Theories der Cognitive Science detailliert beschrieben. Der Vollständigkeit halber sei an dieser Stelle auf eine konträre Position zum Symbolverarbeitungs-Ansatz hingewiesen, den Konnektionismus. Auf diesen neueren Ansatz kann hier aber nicht eingegangen werden. Er wird ebenfalls in Boden (1988) behandelt.

deduktiv-nomologischen Erklärungsbegriff der analytischen Wissenschaftstheorie. Während Erklärungen im Sinn des deduktiv-nomologischen Erklärungsschemas unter Rückgriff auf *Gesetze* erfolgen, beziehen sich Kognitionswissenschaftler mit ihrem Erklärungsbegriff auf das Wissen, *wie* ein Ding funktioniert, somit explizit auf das Zustandekommen von Verhalten und die *Struktur* der Entität, welches das zu erklärende Verhalten erzeugt. Sie verwerfen das Subsumtions-Modell der Erklärung von Hempel-Oppenheim als unbefriedigend und ersetzen es durch ein *kausal-mechanistisches Erklärungsmodell*. Wir wollen diesen Erklärungsbegriff kurz erläutern.

Wir haben mehrfach erwähnt, daß Computerprogramme durch eine hierarchische Organisation gekennzeichnet sind, bei der komplexe Aufgaben auf immer einfachere Teilaufgaben reduziert werden. Das Verhalten eines ("intelligenten") Programms kann durch Reduktion komplexer Aufgaben in einzelne ("dümmere") Teilaufgaben vollständig verstanden und damit - im Verständnis der Kognitionswissenschaftler - auch *erklärt* werden: "We posit subprocesses to explain the actions of processes, subsubprocesses to explain the actions of subprocesses, and so on, until we reach the level of elementary information processes. Though the action of the whole program may appear to require brilliance ... the processes into which it decomposes at the first level (the 'main' subroutines) require only moderate brightness. As we descend through the levels of decomposition in the explanation, the spark of intelligence required for the processes at each level gradually dims, until we reach the machine language instruction, which are easy to implement mechanically. The ghost is exorcized by gradually reducing it to simple formal operations as we elaborate the explanation" (Stillings et al 1987: 313).

Das Verhalten eines Programms wird also dadurch erklärt, daß man sich auf die Teile, die dieses Phänomen generieren und das Zusammenwirken dieser Teile bezieht. Das Ganze wird damit sukzessive in einfache, fundamentale Funktionsteile zerlegt, die für sich keine intelligenten Verfahren durchführen. Erst das Zusammenwirken dieser Funktionsteile erzeugt intelligentes Verhalten.

Kognitionswissenschaftler fassen den menschlichen Geist als ein analoges Konstrukt wie Computerprogramme auf: der menschliche Geist kann erklärt werden durch die Auffassung einer Person als eines intelligenten Systems, das mit einer Reihe von jeweils weniger intelligenten Subsystemen ausgestattet ist. Jedes dieser Subsysteme nimmt bestimmte Funktionen wahr und deren Ausführung wird wiederum an weitere, noch weniger intelligente Subsubsysteme usw. delegiert (Heyer 1988: 38). Da Prinzipien von Computersystemen mit denen des menschlichen Geistes vergleichbar sind, kann das, was das Verhalten des Computerprogramms erklärt, genauso zur Erklärung kognitiver Phänomene herangezogen werden. Die Bedingung für eine gültige Erklärung lautet nur, daß I/O-Verhalten und simulierte und simulierende Strukturen identisch sind:[27]

"... for an adequat simulation to be an adequate explanation it must be the case *both* that the behaviours available to the machine correspond to the behaviours available to the organism *and* that the processes whereby the machines produces behaviour simu-

[27] Bei anderen Autoren genügt es bereits, wenn nur das I/O-Verhalten identisch ist. Vgl. (Z9).

late the processes whereby the organism does" (Fodor 1968: 136, Hervorhebung Fodor).

Die Erklärung kognitiver Vorgänge ist damit äquivalent zur Generierung von Programmen, die ähnliches Verhalten künstlich erzeugen: "Thinking is to be explained by writing a program for a thinking process" (Newell/Simon 1971: 152). In diesem Sinn sind die Zitate (Z8) und (Z9) zu verstehen. Die Cognitive Science behauptet damit, daß es für die wissenschaftliche Erklärung intelligenten Verhaltens gleichgültig ist, was für ein Ding es ist, das sich intelligent verhält - Mensch oder Computer. Es gibt nur eine - uns derzeit noch unbekannte - Grundtheorie intelligenten Verhaltens, aus der sich für jeden Einzelfall eine Erklärung ableiten läßt. Jedem Intelligenzverhalten liegt eine fundamentale Struktur zugrunde, die mit einer Einheitstheorie der Intelligenz erklärt werden kann (Loeck 1986).

Damit erhebt die Cognitive Science den Anspruch, eine empirische Theorie menschlicher Intelligenz auf der Basis von Computerprogrammen zu liefern. Da die Programme, welche kognitive Funktionen nachbilden, diese gleichzeitig erklären, kann in der Auffassung der starken Variante der Cognitive Science jedes dieser Programme als Theorie über die modellierten Phänomene betrachtet werden. Newell und Simon argumentieren z.B., daß solche Programme in dem gleichen Sinn eine Theorie enthalten wie die Gleichungen von Newtons Dynamik eine Bewegungstheorie des Sonnensystems enthalten. Während die Differentialgleichungen von Newtons Theorie bestimmen, was als nächstes geschehen wird - in Abhängigkeit des exakten Systemzustands zu Anfang eines Intervalls - bestimmt das Programm, was als nächstes geschehen wird, in Abhängigkeit von seinem Zustand - der von der Geschichte und der Umgebung abhängt (Newell/Simon 1971).

Haugeland (1981: 2) fordert, daß eine Theorie der "natürlichen" Intelligenz die gleiche Basisform haben sollte wie eine Theorie, die intelligente Computersysteme erklärt. Der normalen Psychologie würde dann lediglich die Aufgabe zufallen, die von der Cognitive Science akzeptierten Konstrukte auf ihre empirische Brauchbarkeit hin zu prüfen (Kobsa 1986: 116). Dies geht sogar soweit, daß nicht als Computerprogramm formalisierte Theorien von den Theoretikern der Cognitive Science als "folk psychology" abgelehnt werden (Boden 1988). Fodor (1980) behauptet, daß die Computational Psychology im Sinn der Cognitive Science die einzige theoretische Psychologie ist, die wir jemals hoffen können zu erreichen.

Die meisten Kognitionswissenschaftler verstehen die Cognitive Science damit als eine Wissenschaft, die den Wissenschaftsbegriff der analytischen Wissenschaftstheorie sprengt. Kognitionswissenschaftler suchen nach wissenschaftlichen Erklärungen, die nicht-nomologische, kausal-mechanistische Erklärungen sein sollen und wollen auch Explananda, die eindeutig Gegenstand der Naturwissenschaft sind, auf eine andere, nicht-nomologische Weise erklären (Loeck 1986: 16). Selbst Stegmüller (1983: 482) hegt die Vermutung, daß es sich bei der Cognitive Science möglicherweise um eine Wissenschaft *sui generis* handle, also einer Wissenschaft eigener, neuer Sorte, die sich dem nomologischen Erklärungsschema der analytischen Wissenschaftstheorie entzieht. Die Diskussion um den erkenntnis- und wissenschaftstheoretischen Status der Cognitive Science ist seit ihrem Entstehen in vollem Gang und nimmt in der - eher

kognitionswissenschaftlichen als wissenschaftsphilosophischen - Fachliteratur einen breiten Raum ein.[28]

3.2.2 Einwände gegen die Computational Theories der Cognitive Science

Ohne daß hier tiefer auf wissenschaftsphilosophische Fragen der Cognitive Science eingegangen oder die Frage nach dem wissenschaftstheoretischen Status der Disziplin gar entschieden werden kann, sollen zwei Einwände gegen die Theorieauffassung der Cognitive Science angerissen werden.

In ihrem Aufsatz "Ist Simulation Erklärung? Cognitive Science wissenschaftstheoretisch betrachtet" untersucht Loeck (1986) die Frage, ob die Annahme der Cognitive Science gerechtfertigt ist, nach der die Theorie der natürlichen Intelligenz des Menschen grundsätzlich mit der Theorie identisch sei, die das intelligente Verhalten der technisch produzierten Computersysteme erklärt. Etwas präziser und allgemeiner läßt sich diese Behauptung folgendermaßen formulieren (Loeck 1986: 18):

Wenn ein Verhalten V_i eines Dings A durch ein Verhalten V_i* eines Dings B ununterscheidbar (=perfekt) simuliert wird, dann impliziert die Theorie T_i*, die das simulierende Verhalten V_i* unter Bezug auf B erklärt, diejenige Theorie T_i, die das simulierte Verhalten V_i unter Bezug auf A erklärt.

Am Beispiel der Theorie des natürlichen Ferro- und künstlichen Elektro-Magnetismus weist Loeck nach, daß beide Phänomene *nicht durch eine einheitliche Theorie erklärt* werden können. Wie in der Cognitive Science zeigen beide physikalische Phänomene ununterscheidbares Verhalten: jeder natürliche Magnet kann durch einen Elektromagneten simuliert werden. Die Theorie des natürlichen Magnetismus muß jedoch Begriffsentitäten verwenden, die in der Theorie des künstlichen Magnetismus nicht enthalten sind, so daß zwar die Theorie des künstlichen Magnetismus von der Theorie des natürlichen Magnetismus impliziert wird, aber nicht umgekehrt.

Die Simulation liefert damit nicht die Erklärung des simulierten Systems. Das simulierte Verhalten V und das perfekt simulierende Verhalten V* können nicht durch dieselbe Theorie (der Magnetfelderzeugung) erklärt werden. Ein Beispiel ist zwar kein Beweis und keine unmittelbare Widerlegung der Simulation-ist-Erklärung-Annahme, es spricht aber zumindest ein Präzedenzfall aus der Physik dagegen, in dem diese Annahme nachweisbar falsch ist. Loeck sieht deshalb die Behauptung der Cognitive Science als eine unhaltbare Spekulation an.[29]

Kobsa (1982 und 1984) wendet sich gegen die Auffassung, Programme im Umfeld der Cognitive Science als Theorien anzusehen und verweist auf den unterschiedlichen formalen Aufbau von Theorien und Computerprogrammen. Computerwissenschaftler und Kognitionswissenschaftler beziehen sich bei der Erklärung der Fähigkeiten von Programmen auf die Funktionen dieser Module und bedienen sich eines funktionalanalytischen Vokabulars. Ein Computermodell ist eine funktionale Dekomposition: es

---

[28] Vgl. z.B. Dennett (1978), Pylyshyn (1980) und Zeitschriften wie "The Behavioral and Brain Sciences", "Cognitive Science" und "Cognitive Psychology".

[29] Man darf aber nicht übersehen, daß Loeck sich in ihrer Argumentation auf den traditionellen Erklärungsbegriff bezieht. Ändert man den Begriff der Erklärung, so kann die Behauptung "Simulation ist Erklärung" trivial werden.

kann in verschiedene Teile mit spezifischen Funktionen aufgeteilt werden, deren Zusammenwirken das Resultat erzeugt. Die einzelnen Teile können durch funktional äquivalente Teile ausgetauscht werden (vgl. Kap. 2.1, S.22-23), so daß es verschiedene (unter Umständen unendlich viele), funktional äquivalente Rekonstruktionsmöglichkeiten gibt. Ein Computerprogramm wäre dann lediglich eine Instantiierung irgendeiner solchen funktionalen Dekomposition und ist lediglich ein Beweis, daß es möglich ist, ein bestimmtes Verhalten durch effektive Verfahren zu rekonstruieren. Damit hat man aber nur ein künstliches Individuum konstruiert, ein Modell, nicht aber die Theorie selbst. Was in den Aussagen der Kognitionswissenschaftler verwechselt wird, sind die Begriffe "Theorie" und "Modell" (Kobsa 1982: 933). KI-Programme sind nicht Theorien *über* Strukturen, Mechanismen und Prozesse, sondern *sind* diese Strukturen, Mechanismen, Prozesse (Kobsa 1982: 934).

Kobsa stützt seine Argumentation durch Gegenbeispiele wie den mittelalterlichen Uhrwerkmacher, der durch das Bauen von immer komplizierteren Uhrwerkmechanismen niemals die zugrundeliegenden Newtonschen Gesetze entdecken würde. Ähnlich argumentiert T.W. Simon (1979: 238): "It is easy to envisage someone constructing a device, such as a model automobile, which corresponds in every relevant respect to the available behaviours and internal processes of an actual automobile and which nonetheless, cannot provide any universal laws governing the operation of the automobile. This construction might be good technology but not good science". Die Autoren wollen damit illustrieren, daß durch das Bauen immer komplexerer Modelle niemals die Theorie entdeckt wird, welche diese Modelle beschreibt.[30]

In unserem Verständnis sind die Einwände von Kobsa und Simon in Abhängigkeit von der wissenschaftsphilosophischen Position relativ einfach - zumindest prinzipiell - zu entscheiden:

- Angenommen, die Cognitive Science sei eine Wissenschaft sui generis im Stegmüllerschen Sinn - was sich allerdings erst noch erweisen müßte: Dann würden die Einwände nicht zutreffen, da die Cognitive Science nicht nach Gesetzen und Erklärungen im traditionellen Sinn sucht.
- Angenommen, die Cognitive Science hat nicht den Status einer Wissenschaft sui generis mit eigenem Erklärungscharakter und man ordnet Theorien der Cognitive Science dem traditionellen Standardtheorienkonzept zu. Dann sind die Einwände zumindest z.T. berechtigt, da die Programme der Cognitive Science nicht per se als Erklärungen und Theorien fungieren können. Ein Programm ist nicht zwangsläufig eine Theorie, nur weil es I/O-äquivalente Ausgaben erzeugt.

Aber auch in diesem Fall - wenn man als Kriterium das Vorliegen gesetzesartiger Regularitäten akzeptiert - ist die von Kobsa angesprochene Funktionssichtweise *kein grundsätzlicher* Einwand dagegen, daß Programme Theorien repräsentieren können. Interpretiert im Sinn des Aussagenkonzepts ist vielmehr entscheidend, ob sich Teile als gesetzesartige Regularitäten identifizieren lassen, die in einer relevanten Beziehung zum empirischen System stehen und aus denen sich eine empirische Theorie rekonstruieren läßt. Theoretische Strukturen in Form nomologischer Aussagen können sich

[30] Allerdings sind beide Beispiele schlecht gewählt, da es in keinem Fall eine naturwissenschaftliche Theorie gibt, die solche Systeme behandelt.

experimentell aus, mit und in den Programmen mit fortschreitender Entwicklung realisieren. Da es keinen ernsthaften Einwand gibt, aus einer bestehenden Theorie ein Computerprogramm zu machen und die Gesetze im Programmcode zu repräsentieren, muß es auch im umgekehrten Fall möglich sein - wenn keine explizite Ausgangstheorie vorliegt -, daß sich hier nomologische Prinzipien finden lassen, die ein Fachwissenschaftler als theoretische Aussagen zu den im Modell abgebildeten Phänomenen akzeptieren würde.

Unabhängig von der Frage, ob mit dem Programm wirklich eine Theorie vorliegt, können Programme zweifellos durch ständiges Experimentieren, Verbessern und Verfeinern theoretische Einsichten liefern und die Bildung von Theorien fördern, so daß auf der Basis von Computerprogrammen zumindest Theorien konstruiert werden können. In der kognitiven Psychologie wird die Theoriebildung beispielsweise entscheidend von Computermodellen angeregt. Theorien über Schemata oder propositionelle Repräsentationen verdanken ihr Entstehen im wesentlichen Computerprogrammen (vgl. Wessels 1984). Dieses Vorgehen ist sicherlich auch außerhalb der kognitiven Disziplinen anwendbar, wie das oben erwähnte TEAMWORK-Modell belegt.

3.2.3 Induktives Entdecken von Gesetzen

Wir möchten abschließend noch auf eine andere Programmklasse hinweisen, die diesem Bottum-Up- oder datenbasierten Ansatz zugerechnet werden kann. Wir haben eben gesagt, daß sich theoretische Strukturen in und aus den Programmen entwickeln können. Diese bewußt vage Sprechweise hat in der KI ein präzises Äquivalent. Unter dem Label "Maschinelles Lernen" gibt es Programme, die (natur-)wissenschaftliche Gesetze induktiv aus Daten ableiten können (z.B. Langley et al 1987, ein Überblick findet sich in Slezak 1989). Gibt man einem solchen Programm z.B. die von Robert Boyle um 1660 benutzten Rohdaten als Input, dann findet das Programm das entsprechende Gesetz von Boyle bezüglich Gasdruck und Volumen als Output (Slezak 1989).

Eine bekannte Familie von solchen Entdeckungsmodellen sind die BACON-Programme, deren Ergebnisse unter Titeln wie "Rediscovering Chemistry with the BACON System" oder "Rediscovering Physics with BACON.3" veröffentlicht wurden. In letzterem beschreibt Langley (1979) eine Programmversion, die empirische Gesetze der Physik unter Nutzung einiger weniger Heuristiken entdeckt: "These rules detect constancies and trends in data, and lead to the formulation of hypothesis and the definition of theoretical terms. BACON.3 represents data at varying levels of description, where the lowest have been directly observed and the highest correspond to hypotheses that explain everything so far observed" (Langley 1979: 505). BACON.3 hat Versionen folgender Gesetze wiederentdeckt: das ideale Gasgesetz, Keplers drittes Gesetz, das Gesetz von Coulomb und Ohm und Galileis Pendelgesetz. Inzwischen gibt es - beginnend mit BACON.1 - eine ganze Reihe unterschiedlich leistungsfähiger Versionen.

Summarisch lassen sich diese Entdeckungsprogramme wie folgt beschreiben: sie nehmen als Input eine Menge von Beobachtungsdaten und versuchen eine Menge von allgemeinen Gesetzen zu formulieren, die diese Daten bündeln. Zur Entdeckung von Invarianten in empirischen Daten oder mathematischen Konzepten werden bestimmte Heuristiken und Problemlösetechniken der KI verwendet. All diesen Programmen unterliegt die Annahme, daß wissenschaftliches Entdecken eine Spezialform von Problemlösen ist (Langley et al 1987: 13).

Die Entdeckungsmechanismen sind weder beschränkt auf quantitative noch auf empirische Gesetze. Beispielsweise entdeckt GLAUBER qualitative Gesetze aus der Chemie über Säuren und Basen, wie z.B.: "Säuren reagieren mit Basen zu Salz" (Langley et al. 1987). Ein anderes Programm, AM (Lenat, vgl. Slezak 1989), hat wichtige Konzepte der Zahlentheorie wiedergefunden, wie z.B. bestimmte Eigenschaften der natürlichen Zahlen, die vier arithmetischen Grundoperationen oder die Goldbachsche Vermutung, nach der jede gerade Zahl als Summe zweier Primzahlen dargestellt werden kann.

Die Möglichkeit der Implementation solcher Programme zeigt, daß die Entdeckungsheuristiken nicht von spezifischen Eigenschaften des Problembereichs abhängen, sondern allgemein auf viele unterschiedliche Domänen anwendbar sind (Slezak 1989: 569). Die Programme deuten somit darauf hin, daß es formale Entdeckungsregeln und ein rationales Induktionsprinzip *gibt* (Slezak 1989: 567). Da der Prozeß des Entdeckens von Theorien genauso rational zu rechtfertigen sei wie die Falsifikation, setzen sich Langley et al (1987) für die Möglichkeit einer psychologischen und normativen Theorie der Entdeckung ein. Dies kontrastiert zur Auffassung von Popper, nach der das Humesche Induktionsprinzip und damit die *Entdeckung* von Gesetzen und Theorien rational nicht zu rechtfertigen sei. Es bleibt abzuwarten, ob diese Programme Ausgangsbasis einer "Theorie der induktiven Entdeckung" sein werden, welche die Ideen von Bacon und Mill präzisieren.

In jedem Fall bestätigen die Programme die Nützlichkeit von datengesteuerten induktiven Entdeckungsmethoden. Prinzipiell wären solche Programme auch in den Sozialwissenschaften wünschenswert und nötig. Inwieweit induktives Herleiten von Regularitäten in sozialwissenschaftlichen Daten aber *möglich* ist, wagen wir hier nicht zu entscheiden. Solche Systeme müßten jedenfalls die Fähigkeit haben, statistische Gesetzmäßigkeiten zu abstrahieren. Einen ersten Anfang macht Garson (1987), der einen Klassifikations- und Regelgenerierungsalgorithmus auf historische Datenmatrizen anwendet.

4. Wissensverarbeitung und Modellbildung

Wir beschränken uns im folgenden weitgehend auf den theoriegesteuerten Ansatz und gehen davon aus, daß Computermodelle auf der Basis bereits vorliegender Theorien konstruiert werden. Wir haben oben festgestellt, daß die konventionelle Computermodellierung im allgemeinen und die Transformation von Theorien in Programme im besonderen mit zum Teil gravierenden Schwierigkeiten und Mängeln verbunden ist, die wir hier noch einmal zusammenfassen wollen:

- Im Verständnis der Mainstream-Simulierer ist die zu simulierende Theorie quantitativ und in Gleichungen zu modellieren. Strukturelle und qualitative Theorien entziehen sich damit weitgehend der Computermodellierung;
- Das I/O-Verhalten von traditionellen Computermodellen ist undurchsichtig, man weiß nicht, aufgrund welcher Prinzipien ein Output zustande kommt;
- Theorien enthalten Propositionen, Computerprogramme Algorithmen, oder: der Theoretiker denkt deklarativ, der Computermodellierer prozedural. In konventionellen Modellen ist die semantische Lücke zwischen qualitativer Theorie und Programm beträchtlich;
- In traditionellen Programmen existiert keine klare Schnittstelle zwischen theoretisch relevanten und theoretisch nicht relevanten Programmteilen, so daß eine Trennung nur schwer durchzuführen ist.

Der nun zu behandelnde Ansatz der wissensbasierten Modellierung behebt oder reduziert zumindest diese Nachteile und bietet eine mächtige Alternative zu traditionellen Modellierungsansätzen. Wir gehen so vor, daß wir zunächst die wichtigsten Aspekte wissensbasierter Systeme unabhängig von der Modellierungsfrage vorstellen und dann deren Anwendung auf wissenschaftliche Theorien behandeln.

4.1 Wissensbasierte Systeme

In der KI-Forschung begann Mitte der sechziger Jahre eine Bewegung die versuchte, naturwissenschaftliches und anderes Wissen so aufzubereiten, daß Computer damit arbeiten konnten. Initiiert von Edward Feigenbaum, Bruce Buchanan und ihren Kollegen von der Stanford University bildet diese Strömung seit Mitte der siebziger Jahre ein Hauptanliegen der Künstlichen Intelligenz und ist inzwischen als ein Gebiet etabliert, das als Wissensverarbeitung bekannt ist. Zu den populärsten Produkten des Wissensverarbeitungsansatzes - und damit der Künstlichen Intelligenz überhaupt - gehören wissensbasierte Systeme. Sowohl in der Praxis als auch in der Wissenschaft werden diese Systeme auf vielfältige und nutzbringenden Weise eingesetzt. Die grundlegenden Eigenschaften von wissensbasierten Systemen sollen nun skizziert werden.

4.1.1 Allgemeine Charakterisierung

Wissensbasierte Systeme sind eine relativ neue Klasse von Computerprogrammen, die das Wissen eines bestimmten Fachgebiets auf einem Rechner verfügbar machen. Der Computer wird als eine allgemeine und einfach zugängliche Wissensquelle betrachtet, die *Wissen verarbeitet*, selbständig *Schlüsse zieht* und *Folgerungen erklärt*. Der Begriff "wissensbasiert" soll dabei ausdrücken, "daß ein solches System über eine Wissensbasis verfügt, in der das vom System benutzte Wissen explizit codiert ist - also nicht implizit im Programmcode 'versteckt' ist" (Reimer 1991: 2-3). Wissen legen wir einfach pragmatisch fest als Menge aller für wahr angenommenen Aussagen über einen repräsentierten Weltausschnitt durch einen "Wissensträger" (Reimer 1991: 6).

Zur Lösung ihrer Aufgaben benötigen wissensbasierte Systeme detaillierte Kenntnisse über das Aufgabengebiet und Strategien, *wie* dieses Wissen benutzt werden soll. Zur Verwendung des Wissens in der Maschine muß dieses deshalb *formalisiert*, im Computer *repräsentiert* und gemäß einer Problemlösungsstrategie *manipuliert* werden (Puppe 1988: 2). Wissensformalisierung, -repräsentation und -manipulation geschieht dabei unter Verwendung von Methoden der KI.

Im Kontext wissensbasierter Systeme ist der relevante Wissensträger in der Regel ein menschlicher Experte, also eine Person, die fundierte Fachkenntnisse auf einem Spezialgebiet besitzt. Diese Systeme enthalten damit das Spezialwissen von Experten und sollen ein Problemlösungsverhalten zeigen, das dem eines Experten vergleichbar ist. Da solche Systeme den Lösungsprozeß von Experten "simulieren", ist für diese Programme auch der Ausdruck "Expertensysteme" üblich.[31] Typische Anwendungen sind medizinische Diagnostik, Lokalisierung von Fehlern in technischen Systemen oder Interpretation von gemessenen Daten.

Das Wissen von Experten unterscheidet sich grundlegend von "Wissen", wie es in konventionellen Computerprogrammen enthalten ist[32] und läßt sich durch folgende drei Merkmale charakterisieren:

- Expertenwissen ist im allgemeinen nicht algorithmisch und erst recht nicht numerisch, vielmehr besteht es aus symbolischen Beschreibungen, welche die empirischen Beziehungen in einem Problembereich charakterisieren.
- Expertenwissen ist oft vage und unsicher, Zusammenhänge können nicht genau angegeben werden, häufig werden "Daumenregeln" benutzt, die nicht immer, aber

---

[31] Manche Autoren machen einen Unterschied zwischen den Begriffen "Expertensystem" und "wissensbasiertes System". Für Bratko (1987: 342) beispielsweise sind Expertensysteme spezielle wissensbasierte Systeme, die über eine Erklärungskomponente (vgl. 4.1.2 Architektur von Expertensystemen) verfügen. Wissensbasierte Systeme müßten danach nicht über eine solche Komponente verfügen. Andere Autoren fassen umgekehrt wissensbasierte Systeme als Teilmenge von Expertensystemen auf. Für Harmon/Maus/Morrissey (1989: 19) sind wissensbasierte Systeme "kleine" Expertensysteme, die nur eine Teilmenge von Techniken verwenden, wie sie in großen Systemen eingesetzt werden. Da bei den später implementierten Modellen ebenfalls nur elementare Techniken eingesetzt werden, bevorzugen wir den Begriff des "wissensbasierten Systems", betrachten beide Terme aber im weiteren Verlauf der Arbeit als synonym.

[32] Auch numerische Programme beinhalten "Wissen", z.B. enthält jedes konventionelle Statistikprogramm numerische Algorithmen und damit "Wissen" zu statistischen Verfahren. Wie eingangs erwähnt, soll der Term "wissensbasiert" aber ausdrücken, daß das Wissen *explizit* codiert ist.

meist zum Erfolg führen.
- Experten verfügen über einen Wissenskörper, der unabhängig davon ist, wie dieser bearbeitet wird.

Da konventionelle Programme exakte, numerisch orientierte Algorithmen voraussetzen, in denen keine klare Trennung von Ablaufsteuerung und inhaltlichem Wissen möglich ist, eignen sich diese nicht zur Repräsentation von Expertenwissen. Entsprechend unterscheiden sich Expertensysteme von konventionellen Computerprogrammen vor allem in diesen drei Punkten, die als die Schlüsselideen der wissensverarbeitenden KI angesehen werden können (Harmon/Maus/Morrissey 1989: 19-21):
- Expertensysteme *repräsentieren Wissen*, verwenden also symbolische Ausdrücke, während konventionelle Programme vorrangig numerische Ausdrücke und Verfahren benutzen.
- Expertensysteme sind *heuristisch ausgerichtet* und eignen sich für diffuse Bereiche, was sie insbesondere für Anwendungsgebiete interessant macht, die unsicher, vage und unvollständig sind.
- Expertensysteme *trennen Wissen und Problemlösungsstrategie,* während in konventionellen Programmen beides vermischt wird.

Insbesondere der letzte Punkt - die Trennung des Wissens von der Bearbeitung des Wissens - wird als das entscheidende Charakteristikum und *grundlegende Organisationsprinzip von Expertensystemen* betrachtet (vgl. Puppe 1988: 2-3). Diese Trennung wird begünstigt durch eine Darstellung des Wissens in Form von Wenn-dann-Regeln. Eine Menge solcher Wenn-dann-Regeln zusammen mit bestehenden Fakten nennt man *Wissens- oder Datenbasis*. Diese Wissensbasis wird bearbeitet von einem *Inferenzmechanismus,* der *unabhängig* und *getrennt* von ihr existiert. Der Inferenzmechanismus übernimmt die Abarbeitung der Regeln und generiert - eventuell unter Konsultation des Benutzers - neue Fakten. Er *manipuliert* also das Fachwissen, so daß neues Wissen abgeleitet werden kann.

Die Wissensdarstellung in Form von Wenn-dann-Regeln bedingt insbesondere die folgenden Eigenschaften von Expertensystemen (vgl. Puppe 1988: 4, Bratko 1987: 345):
- *Modularität*: Jede Regel definiert ein kleines, unabhängiges Stück Information;
- *Flexibilität*: Einzelne Wissensstücke (Regeln) können relativ leicht hinzugefügt, verändert oder gelöscht werden;
- *Transparenz*: Expertensysteme können ihre Problemlösung durch Angabe des benutzten Wissens erklären.

Mit "Erklären" ist gemeint, daß das System seine Anfragen und Entscheidungen rechtfertigen kann und insbesondere Auskunft gibt, *warum* es eine bestimmte Frage stellt, und *wie* es eine bestimmte Folgerung erreicht hat. Der Zweck von Erklärungen liegt in der Plausibilitätskontrolle, der Nachvollziehbarkeit und Transparenz des Lösungswegs, sowie im Nachweis der Korrektheit der vorgeschlagenen Lösung.

Darüber hinaus ist die Darstellung des Expertenwissens in Form von Wenn-dann-Regeln beliebt, weil diese eine natürliche und häufig verwendete Form der Wissensmitteilung sind. "Da fast jeder Experte auf Anhieb einige Regeln aus seinem Fachgebiet nennen kann und durch die Verknüpfung schon relativ weniger Regeln oft

eine erstaunliche Leistungsfähigkeit erreicht wird, ist der regelbasierte Programmierstil für Expertensysteme sehr populär geworden" (Puppe 1988: 3).

Die Terme "wissensbasiert" und "regelbasiert" werden deshalb oft synonym als Bezeichnung für diese Klasse von Programmen verwendet, und der Inferenzmechanismus wird in diesem Zusammenhang als "Regelinterpreter" bezeichnet. Weil regelbasierte Systeme aus vorhandenem Wissen mit Regeln neues Wissen *produzieren*, sind auch die Terme "Produktionsregel" und "Produktionssystem" üblich.

4.1.2 Architektur

Unter der Architektur von Computerprogrammen versteht man die verschiedenen Programmteile und ihre Beziehungen untereinander. Entsprechend der funktionalen Trennung von Expertenwissen und Inferenzmechanismus bilden

- die Wissens- oder Datenbasis und
- der Inferenzmechanismus

die minimalen Teile und Hauptkomponenten von Expertensystemen. Der Inferenzmechanismus kann dabei als Teil eines allgemeineren Steuersystems aufgefaßt werden. Zur systematischeren Darstellung der Architektur beziehen wir uns im folgenden auf Puppe (1988: 12-13).

Die erste Hauptkomponente bildet das Steuersystem mit folgenden Teilkomponenten:

- Der *Inferenzmechanismus* (auch: Inferenz- oder Schlußfolgerungskomponente/-maschine, bei regelbasierten Systemen: Regelinterpreter) ist Programmcode, der aus dem zur Verfügung stehenden Wissen *neues Wissen ableitet*. Die Inferenzmaschine sucht und verknüpft Fakten und Regeln nach einer vorgegebenen Strategie und produziert so Folgerungen und neue Ergebnisse. Die Funktionsweise dieser Inferenzmaschine ist abhängig von der gewählten Wissensrepräsentation und wird weiter unten näher behandelt.
- Die *Interviewerkomponente* führt den interaktiven Dialog mit dem Benutzer. Der Dialog sollte sich nach den Erfordernissen des Endbenutzers richten.
- Die *Erklärungskomponente* macht die Vorgehensweise und die innere Struktur des Expertensystems transparent. Sie zeigt dem Benutzer, durch welche Fakten und Regeln eine Folgerung zustande kam. Dem Experten gibt sie die Möglichkeit zu überprüfen, ob die Schlußfolgerungen des Systems inhaltlich korrekt sind. Man unterscheidet direkte und indirekte Erklärungen. *Indirekte Erklärungen* bestehen aus einfachen Texten, mit denen der Programmierer Teile seines Programms explizit kommentiert. *Direkte Erklärungen* sind eine Aufbereitung der Problemlösungsschritte ("Trace"), die zu dem zu erklärenden Ergebnis geführt haben und werden *direkt aus dem Programmcode* abgeleitet. In den direkten Erklärungen liegt der eigentliche Fortschritt, da sie zeigen, *welche Teile des Programms zur Herleitung eines Ergebnisses notwendig waren.*
- Die *Wissenserwerbskomponente* sollte dem Experten die Möglichkeit geben, sein Wissen in das System einzugeben und später wieder zu ändern.

Die zweite Hauptkomponente eines Expertensystems bildet die Wissensbasis.

- Die *Wissens- oder Datenbasis* enthält die Kenntnisse des Fachmanns, meist in Form von Fakten und Regeln, sowie das dynamische, austauschbare Wissen des Benutzers. Puppe (1988) unterscheidet nach der *Herkunft des Wissens* genauer drei Wissensarten:
  - das *bereichsbezogene (Regel-)Wissen* des Experten (zeitlich langfristig gültig),
  - das *fallspezifische Wissen* vom Benutzer (gültig nur für die Dauer einer Sitzung), sowie
  - *Zwischen- und Endergebnisse* (ebenfalls nur für eine Sitzung), die von der Inferenzmaschine hergeleitet wurden.

  Eine andere Gliederung des Wissens orientiert sich am *Gebrauch des Wissens*. Die daraus resultierenden Wissensarten sind *Faktenwissen*, *Ableitungswissen* und *Kontroll- oder Steuerungswissen*.

Mit dieser Unterteilung kann die Architektur eines Expertensystems grafisch wie in Abb. 4.1 dargestellt werden.

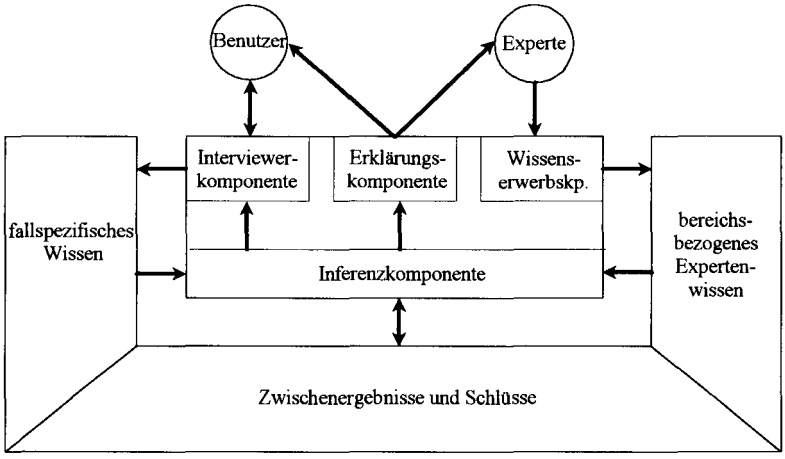


Abb. 4.1: Architektur eines wissensbasierten Systems (nach Puppe 1988: 13).
Die Pfeile sollen hierbei den Informationsfluß andeuten. Beispielsweise gibt der Benutzer über die Interviewerkomponente fallspezifisches Wissen ein, der Inferenzmechanismus verarbeitet das Wissen und liefert Schlußfolgerungen über die Interviewerkomponente wieder an den Benutzer zurück.

4.1.3 Nutzen und Anwendungsgebiete

Expertensysteme stellen die Kompetenz von Experten vervielfacht und dezentralisiert zur Verfügung (Savory 1988: 7). Benutzer kann grundsätzlich der Experte selbst sein, für dessen Fachgebiet das System entwickelt wurde oder ein Laie. Der vordergründigste Nutzen solcher Systeme liegt darin, daß Experten bei Routinetätigkeiten entlastet werden und einfache Probleme auch ohne Experten gelöst werden können (Puppe 1988: 5).

Savory (1988: 12) nennt u.a. folgende Vorteile, die sich beim Einsatz von Expertensystemen ergeben:

- Expertensysteme können dazu eingesetzt werden, den *Experten selbst in seiner Arbeit zu unterstützen und zu entlasten*, wodurch dieser für andere Aufgaben freigestellt wird.
- Hochspezialisiertes *Wissen wird zeitlich und räumlich unbegrenzt verfügbar* und reproduzierbar. Expertenwissen wird gesichert, vervielfältigt und verteilt und kann so einem größeren und inkompetenteren Benutzerkreis zur Verfügung gestellt werden.
- Expertensysteme können mit ihren Problemlöse- und Erklärungsfähigkeiten ideal für *Ausbildungszwecke* eingesetzt werden.
- Expertensysteme lassen sich bei wachsendem Wissensangebot *nicht durch die Fülle der Daten verwirren*, sondern werden im Gegenteil dadurch eher besser.
- Expertensysteme *überprüfen alle relevanten Hypothesen* und sind nicht wie menschliche Experten vergeßlich.
- Im Gegensatz zu konventionellen Programmen können Expertensysteme Auskunft darüber geben, *welchen Lösungsweg* sie einschlagen und *warum* sie *einen bestimmten Lösungsweg* gewählt haben. Dies erhöht die Akzeptanz und das Verständnis solcher Systeme.

Je nach bearbeitetem Problem und verwendeter Problemlösungsstrategie lassen sich Expertensysteme verschiedenen Problemlösungstypen zuordnen. Puppe (1988: 10) unterscheidet drei grundlegende Problemlösungstypen: Diagnostik, Konstruktion und Simulation. In der *Diagnostik wählt* das Expertensystem die Lösung aus einer Menge vorgegebener Alternativen *aus* (Beispiel: Fehlerdiagnose einer Maschine), in der *Konstruktion* wird die Lösung aus kleinen Bausteinen *zusammengesetzt* (Beispiel: Stundenplanerstellung) und in der *Simulation* werden aus *Ausgangszuständen Folgezustände* hergeleitet (Beispiel: Vorhersage von Therapieeffekten). Die meisten existierenden Expertensysteme sind dem ersten Typ zuzurechnen.

Hauptanwendungsgebiete von Expertensystemen sind Technik, Medizin und Naturwissenschaften. Bekannte Expertensysteme sind MYCIN (Medizin, Diagnose und Therapie von Infektionskrankheiten), PROSPECTOR (Geologie, Gesteinsanalyse), DENDRAL (Chemie, Analyse organischer Molekülstrukturen), MOLGEN (Biologie, Planung genetischer Experimente) und MACSYMA (Mathematik, symbolische Lösung von Differentialgleichungen, unbestimmten Integralen, Vektoralgebra).

Eine ausführliche Beschreibungen einzelner Expertensysteme und ein zusammenfassender Überblick findet sich in Hayes-Roth/Waterman/Lenat (1983), Harmon/King (1986) und Waterman (1986).

4.1.4 Ein Beispiel: MYCIN

Die vorgestellte allgemeine Darstellung von Expertensystemen soll durch einen typischen Dialogausschnitt mit *dem* "Klassiker" unter den regelbasierten Systemen - dem eben erwähnten MYCIN - verdeutlicht werden.

MYCIN (Shortliffe 1976) ist ein an der Stanford University entwickeltes, frühes und sehr bekanntes medizinisches Expertensystem zur Diagnose und Therapie bakterieller Infektionskrankheiten des Blutes und der Gehirnhaut. Es sammelt Infor-

mationen zu Symptomen des Patienten, seiner Anamnese und bestimmten Labortests. Die Informationen werden interaktiv durch Fragen an den Benutzer eingeholt. Eine von MYCIN gestellte Frage ist dabei zu jedem Zeitpunkt bestimmt durch die aktuelle Hypothese und durch alle auf vorhergehende Fragen erfolgten Antworten (Winston 1987: 210).

In dem folgenden, mit Kommentaren ergänzten Protokollausschnitt eines MYCIN-Dialogs (Shortliffe 1976) erscheinen Benutzereingaben immer nach "**".

```
----- PATIENT-1 ------
1)   Patient's Name: (first-last)
**   JEREMIAH SAMPLE
2)   Sex:
**   MALE
3)   Age:
**   60

I will refer to the most recent positive culture for which you
desire therapeutic advice as:
----- CULTURE-1 ------

4)   From what site was the specimen for CULTURE-1 taken?
**   BLOOD
5)   How many days ago was this blood culture (CULTURE-1) obtained?
**   2
...
22)  Was any chemotherapy recently begun specifically to combat
     ORGANISM-1?
**   NO
23)  Is Jeremiah Sample (PATIENT-1) a burn patient?
**   RULE

RULE047
IF:
1) THE SITE OF THE CULTURE IS BLOOD, AND
2) THE IDENTITY OF THE ORGANISM IS NOT KNOWN WITH CERTAINTY, AND
3) THE STAIN OF THE ORGANISM IS GRAMNEG, AND
4) THE MORPHOLOGY OF THE ORGANISM IS ROD, AND
5) THE PATIENT HAS BEEN SERIOUSLY BURNED
THEN:
THERE IS WEAKLY SUGGESTIVE EVIDENCE (.4) THAT THE IDENTITY OF THE
ORGANISM IS PSEUDOMONAS

** NO
```

Kommentar: Das System fragt zunächst die Patientendaten ab. Die Eingabe des Schlüsselwortes RULE bei Frage 23) veranlaßt das System, die Regel auszugeben, deren Konklusion es gerade versucht, zu beweisen. In diesem Fall wird gerade Bedingung 5) von Regel 47 geprüft. Die Zahl im Then-Teil der Regel steht für einen sog. Sicherheitsfaktor der ausdrücken soll, daß die Regel nur "bis zu einem gewissen Grad" gültig ist. Damit soll der eingangs erwähnten Tatsache Rechnung getragen werden, daß Expertenwissen oft unsicher und vage ist. In MYCIN wird jeder Regel eine Zahl zwischen 0 (sicher falsch) und 1 (sicher wahr) zugeordnet, welche die

Sicherheit der in der Regel enthaltenen Folgerung darstellt. Die Zahlen ähneln Wahrscheinlichkeiten, sind aber nicht identisch mit ihnen (vgl. Rich 1988: 310).[33]

```
24)  Is the patient's illness with ORGANISM-1 a hospital-acquired
     infection?
**   YES
25)  Do you consider Jeremiah Sample (PATIENT-1) to be a compromised
     host?
     (alcoholic, immunosuppressed, etc.)
**   NO
26)  Has Jeremiah Sample had any gu surgery or manipulation of the
     urinary tract(e.g., indwelling catheter, cystoscopy, etc.)
**   WHY

I AM TRYING TO FIND OUT ABOUT THE IDENTITY OF THE ORGANISM.
I ALREADY KNOW THAT:
     THE PORTAL OF ENTRY OF THE ORGANISM IS UPPER-GU
     THE MORPHOLOGY OF THE ORGANISM IS ROD
     THE STAIN OF THE ORGANISM IS GRAMNEG
     THE SITE OF THE CULTURE IS BLOOD
     THEREFORE,

IF:
1)  THE PATIENT HAS NOT HAD A GENITO-URINARY MANIPULATIVE PROCEDURE
    AND
2)  URINARY-TRACT-INFECTION IS NOT A PROBLEM FOR WHICH THE PATIENT
    HAS BEEN TREATED
THEN:
    THERE IS SUGGESTIVE EVIDENCE (.6) THAT THE IDENTITY OF THE
    ORGANISM IS E.COLI
(RULE156)

**   YES
```

Kommentar: Wird statt der Antwort auf eine Frage das Schlüsselwort "WHY" eingegeben, so nennt MYCIN den Grund, warum die Frage gestellt wurde. Dabei gibt es zuerst die aktuell geprüfte Hypothese an, listet alle bereits bekannten Fakten auf, die es zur Herleitung der Hypothese braucht und gibt anschließend die entsprechende Regel aus, welche die Hypothese zur Folge hat.

Nach weiteren Fragen wird schließlich eine Diagnose gestellt und - unterstützt durch zusätzliche Informationsanfragen - eine Empfehlung für eine Behandlung vorbereitet.

```
My therapy recommendation will be based on the following possible
identities of the organism(s) that seem to be significant:
<Item 1> The identy of ORGANISM-1 may be PSEUDOMONAS
<Item 2> The identy of ORGANISM-1 may be KLEBSIELLA
<Item 3> The identy of ORGANISM-1 may be ENTEROBACTER
<Item 4> The identy of ORGANISM-1 may be E.COLI
<Item 5> The identy of ORGANISM-1 may be PROTEUS
```

[33] Die Behandlung von Unsicherheit ist von Expertensystem zu Expertensystem unterschiedlich. Ein Überblick über die wichtigsten Modelle und deren Kritik findet sich in Puppe (1988).

```
48) On a scale of 0 to 4, where higher numbers indicate increasing
    severity, how would you rate Jeremiah Sample (PATIENT-1)'s
    degree of sickness?
**  4
49) Does Jeremiah Sample (PATIENT-1) have a clinically significant
    allergic reaction to any antimicrobial agent?
**  NO
[Considering GENTAMICIN for use against ORGANISM-1]
...
My preferred therapy recommendation is as follows:
In order to cover for items <1> <2> <3> <4> <5>:
Give the following in combination:
1 GENTAMICIN
Dose: 1.7 MG/KG Q8H-IV OR IM
Comments: MODIFY DOSE IN RENAL FAILURE
2. CARBENICILLIN
Dose: 25MG/KG Q2H-IV
Comments: MODIFY DOSE IN RENAL FAILURE
```

Die Erklärungsoption "HOW" (wie ist eine Schlußfolgerung hergeleitet worden), die in diesem Dialog nicht erscheint, wird mit der Aufzählung aller Regeln beantwortet, deren dann-Komponente die Schlußfolgerung enthält.

Die Wissensbasis von MYCIN besteht aus ca. 500 Wenn-dann-Regeln der obengenannten Art, die rückwärtsverkettet abgearbeitet werden (siehe unten). Das System stellte bei durchgeführten Tests Diagnosen, die in 90% mit denen von Spezialisten übereinstimmten. Die Leistungsfähigkeit des Systems liegt über den Fähigkeiten eines normalen Hausarztes und wird nur von einzelnen universitären Spitzen übertroffen. MYCIN ist ausführlich beschrieben in Shortliffe (1976) und Harmon/King (1986).

4.2 Wissensrepräsentation

Die Repräsentation von Wissen mit Wenn-dann-Regeln - wie in MYCIN - ist ein weit verbreiteter Formalismus, den wir zusammen mit logischen Darstellungsformen in den nächsten Kapiteln vorstellen wollen. Zunächst sind jedoch einige allgemeine Bemerkungen zu "Wissen" und "Wissensrepräsentation" angebracht.

Ohne ausführlich auf den Terminus "Wissen" einzugehen (vgl. hierzu z.B. Reimer 1991 und Heyer/Krems/Görz 1988), begnügen wir uns mit einigen Stichworten zu verschiedenen Wissenskategorien und Repräsentationsformalismen. In Zusammenhang mit Expertensystemen wurde oben bereits eine Unterteilung nach *Herkunft* und *Gebrauch* des Wissens erwähnt. Eine andere Unterscheidung stammt von Barr/Feigenbaum (1981: 144), die folgende *Wissensarten* festlegen:

1. Wissen als Fakten (Eigenschaften, Beziehungen) über Objekte der Realität;
2. Wissen als Kenntnis über Handlungen und Ereignisse;
3. Wissen über bestimmte Fertigkeiten, Vorgehensweisen und Methoden;
4. Meta-Wissen, d.h. Wissen darüber, was wir wissen.

Die Wissensarten (1) und (3) in der Einteilung von Barr/Feigenbaum entsprechen der in der KI/Informatik üblichen Unterscheidung zwischen deklarativem und prozeduralem Wissen, welche bereits in Kap. 3 erwähnt wurde. Bei *deklarativem Wissen* han-

delt es sich um Tatsachen- oder Faktenwissen, mit dem Sachverhalte ausgedrückt werden, wie etwa bei:

```
(F1) Expertensysteme sind eine spezielle Klasse von Computer-
     programmen.
```

Prozedurales oder Verfahrens-Wissen hat die Form von Algorithmen und kann in Gestalt von Prozeduren dargestellt werden (vgl. z.B. Rechenberg 1991: 227). Das Vorhaben, eine wissenschaftliche Arbeit zu schreiben, kann prozedural wie folgt aufgefaßt werden:

```
(F2) Um eine wissenschaftliche Arbeit zu schreiben, überlege dir ein
     Thema, sammle Literatur zum Thema ...
```

Wissens*repräsentation* bezeichnet das Aufschreiben von Symbolen, die in einer erkennbaren Weise einem Ausschnitt einer zu repräsentierenden Welt entsprechen (Reimer 1991: 9). Ein mächtiges und flexibles Mittel zur Repräsentation von Wissen ist die natürliche Sprache, die sich aber nicht zur Rechnerimplementierung eignet (vgl. Reimer 1991: 28-29). Maschinelle *Wissensrepräsentationsmethoden* beschreiben und erstellen Verfahren, das (Experten-)Wissen im Computer zu erfassen und in geeigneter Form zu nutzen.[34] Die wichtigsten maschinellen Repräsentations*formen* sind:

- prädikatenlogische Formeln,
- Regeln,
- semantische Netze
- Objekte/Frames.

Diese Darstellungsverfahren sind überblicksartig beschrieben in Rich (1988), Tanimoto (1990), Winston (1987) und ausführlich in Reimer (1991). In unserem Rahmen genügen Prädikatenlogik und Regeln, so daß wir uns im folgenden darauf beschränken.

4.2.1 Regelbasierte Wissensrepräsentation

Wie oben erwähnt, sind in Expertensystemen Produktionsregeln die verbreitetste Wissensrepräsentationsmethode, da Experten ihr Wissen oft in Regeln formulieren. Das eben vorgestellte MYCIN-System ist ein Beispiel für eine solches Programm, in dem medizinisches Wissen in Regelform ausgedrückt wird. In den Regeln 47 und 156 des MYCIN-Dialogs wird z.B. Fachwissen über die Identität von Organismen codiert.

Produktionsregeln haben grundsätzlich die Form "Wenn x, dann y". Der obengenannte deklarative Wissensausschnitt (F1) läßt sich etwa schreiben als:

```
(F1') Wenn etwas ein Expertensystem ist, dann ist es ein Computer-
      programm,
```

und der prozedurale Wissensausschnitt (F2) als:

---

[34] Wissensrepräsentationsmethoden sind zu unterscheiden vom konkreten Prozeß der Transformation des Expertenwissens in das Computerprogramm, womit sich das sog. "Knowledge Engineering" befaßt.

```
(F2') Wenn eine wissenschaftliche Arbeit geschrieben werden soll,
      dann überlege dir ein Thema, sammle Literatur ....
```

Sowohl deklaratives als auch prozedurales Wissen läßt sich also in Regelform darstellen, wobei deren Interpretation oder Lesart aber variiert:

- Bei logischer oder *deklarativer* Lesart sind Regeln als *logische Ableitungsregeln* aufzufassen und Wenn-dann-Regeln sind zu lesen als:
  wenn *Bedingung(en)* dann *Konklusion*.
  Ein so interpretiertes regelbasiertes System ist ein *Deduktionssystem*. In diesem Fall kann der Wenn-Teil als *Antezedens* oder *Bedingung*, der Dann-Teil als *Sukzedens, Konklusion, Implikation* oder *Konsequenz* bezeichnet werden.
- Bei *prozeduraler* Lesart produzieren Regeln aus Situationen (linke Seite) Aktionen (rechte Seite) und können gelesen werden als:
  wenn *Situation(en)* dann *Aktion(en)*.
  In diesem Fall kann die rechte Seite als *Handlung* oder *Aktion* angesehen werden, mit der ein Zustand verändert wird.

Je nach Semantik kann sogar die *gleiche* Regel prozedural oder deklarativ interpretiert werden, meist ist aber die Interpretation durch die Problemstellung eindeutig determiniert.
Beispielsweise wird die Regel

```
(F3) Wenn Nackensteife und
     hohes Fieber und
     Bewußtseinstrübung
     dann besteht Verdacht auf Meningitis.
```

am besten deklarativ interpretiert, bei der die dann-Komponente als Implikation oder Deduktion aufgefaßt wird.
Hingegen wird die Regel

```
(F4) Wenn Verdacht auf Meningitis besteht,
     dann nimm sofort Antibiotika.
```

besser prozedural interpretiert, bei der die dann-Komponente eine Handlung ist (Puppe 1988: 21).

Ein wissensbasiertes System wird erst interessant durch die *Verkettung mehrerer Regeln*. Durch die Verkettung vieler Regeln können ganze *Regelnetzwerke* entstehen, die mit steigender Regelmenge immer schwerer überschaubar werden. Der Vorteil des Computers ist hier unübersehbar: mechanisches Ableiten von Hand wäre aufwendig und mit großer Wahrscheinlichkeit inkorrekt und unvollständig. Die folgende Regelmenge stellt beispielsweise bereits ein einfaches Regelnetz dar, in welchem die Regeln miteinander über Konklusion-Antezedens verkettet sind:

```
(N1) Wenn a dann b
     Wenn b dann c
     Wenn c dann d ...
```

Neben Regeln, deren Konklusion sich im Antezedens anderer Regeln findet, kann es Regeln mit gleichem Antezedens und unterschiedlicher Konklusion geben:

```
wenn a dann b,
wenn a dann c,
wenn a dann d.
```

Analog können Regeln existieren mit identischer Konklusion und unterschiedlichem Antezedens:

```
wenn b dann a,
wenn c dann a,
wenn d dann a.
```

Anhand dieser Beispiele läßt sich leicht nachvollziehen, daß bei der Wissensverarbeitung die Regeln mit bestimmten Strategien ausgewählt und manipuliert werden müssen. Diese strategische Abarbeitung der Regeln und das "Navigieren durch das Regelnetz" übernimmt der Regelinterpreter. Man kann sich den Regelinterpreter einfach als Steuerungsprogramm vorstellen, welches die Reihenfolge der Abarbeitung lenkt.[35] Grundsätzlich gibt es zwei strategische Alternativen bei der Manipulation von Regeln: Vorwärts- und Rückwärtsverkettung.

Bei der *Vorwärtsverkettung* (*Forward-Chaining*) geht der Regelinterpreter von der vorhandenen Faktenbasis aus, sucht eine Regel, deren Vorbedingungen erfüllt sind und fügt die Konklusion der Faktenbasis hinzu (die Regel "feuert"). Dieser Prozeß wird solange wiederholt, bis keine Regel mehr anwendbar ist. Zur Illustration ein einfaches Beispiel:
Angenommen, in der Faktenbasis steht

```
a
```

und es seien folgende Regeln gegeben:

```
(R1)  wenn c dann d
(R2)  wenn a und d dann f
(R3)  wenn a dann c
(R4)  wenn g dann j
```

Angenommen weiter, wir wären daran interessiert, ob `f` gilt. Ein vorwärtsverkettender Regelinterpreter würde folgende Fakten produzieren und sie der Faktenbasis hinzufügen (in Klammern die verwendeten Regeln):

```
c (R3)
d (R1)
f (R2)
```

`f` konnte also, unter Verwendung von zwei weiteren Regeln, abgeleitet werden. Da oft mehrere Regeln auf eine bestimmte Faktenmenge anwendbar sind, muß eine Konfliktlösungsstrategie die Auswahl der Regeln übernehmen. Solche Strategien können sein (Puppe 1988: 23):

[35] In der Terminologie der Wissensverarbeitung beinhaltet die Inferenzmaschine bzw. der Regelinterpreter - meist prozedurales - Meta-Wissen, nämlich Wissen zur Bearbeitung von Wissen.

- Auswahl nach Reihenfolge (die erste oder aktuellste Regel feuert),
- Auswahl nach syntaktischer Struktur (die einfachste oder komplexeste Regel feuert), oder
- Auswahl mittels Zusatzwissen (z.B. Meta-Regeln, die den Auswahlprozeß steuern).

Das Gegenstück zur Vorwärtsverkettung ist die Rückwärtsverkettung. Bei der *Rückwärtsverkettung* (*Backward-Chaining*) geht der Inferenzmechanismus von der zu beweisenden Konklusion aus. Steht die Konklusion nicht in der Datenbasis, werden Regeln gesucht, deren Dann-Teil die Konklusion enthält. Gibt es eine solche Regel, wird versucht, den Wenn-Teil dieser Regel zu beweisen, der als Fakt in der Datenbasis oder wiederum im Dann-Teil einer anderen Regel stehen kann usw. Kann ein Ziel nicht bewiesen werden, kann auch eine Frage an den Benutzer gestellt werden. Auch im rückwärtsverkettenden Fall gibt es Konfliktlösungsstrategien, welche die Regelauswahl steuern.

Ein Beispiel: Angenommen, wir wollen mit den Regeln von (N1) beweisen, daß `c` gilt. `c` steht nicht in der Faktenbasis, aber der Regelinterpeter findet die Regel

```
wenn b dann c.
```

Wir haben nun ein neues Beweisziel `b`, das ebenfalls nicht in der Faktenbasis zu finden ist. Die Inferenzmaschine sucht daher wiederum nach einer Regel, um `b` abzuleiten und findet die Regel

```
wenn a dann b.
```

Unser neues Beweisziel heißt damit `a`. Nehmen wir nun an, es wird keine Regel gefunden, die es ermöglicht, `a` abzuleiten. In diesem Fall wird "`Gilt a?`" als Frage an den Benutzer gestellt. Antwortet der Benutzer mit "ja", so ist damit das ursprüngliche Beweisziel `c` hergeleitet, ansonsten schlägt der Beweisversuch für `c` fehl.

Rückwärtsverkettende Systeme haben den großen Vorteil, daß *zielgerichtet* abgeleitet wird und nicht, wie bei der Vorwärtsverkettung, irrelevante Fakten deduziert werden. Beim vorwärtsverketteten Ableiten mit den Regeln (R1)-(R4) wurden beispielsweise zwei Fakten hergeleitet und in die Wissensbasis eingefügt, die gar nicht benötigt werden. Die rückwärtsverkettende Strategie eignet sich besonders zum gezielten Erfragen noch unbekannter Fakten und findet sich in vielen existierenden Systemen, wie z.B. in MYCIN. Das Backward-Chaining-Prinzip von MYCIN läßt sich in dem obengenannten Dialog leicht erkennen. Im allgemeinen ist das Rückwärtsschließen besser (effizienter), es gibt aber auch Ausnahmen. Zu den Vor- und Nachteilen von Vorwärts- und Rückwärtsverkettung vgl. z.B. Savory (1988: 46ff.) oder Puppe (1988). Neben reinen vorwärts- und rückwärtsverkettenden Systemen gibt es noch Verfeinerungen und Mischformen beider Inferenzstrategien, die in unserem Kontext aber nicht von Bedeutung sind.

4.2.2 Prädikatenlogisch basierte Wissensrepräsentation: PROLOG

Einer der ältesten Repräsentationsformalismen für Wissen ist die Prädikatenlogik erster Ordnung.[36] Sie ist die theoretisch am besten untersuchte Darstellungsmethode und häufig verwendeter Bezugspunkt für andere Wissensrepräsentationen (Puppe 1988: 16). Ihre Attraktivität als Medium zur Wissensdarstellung liegt in der schon eingeführten Syntax und Semantik und der Bereitstellung eines formalen Schlußfolgerungsapparats. Die Prädikatenlogik kann nicht nur als abstrakter Repräsentationsformalismus benutzt werden, sondern sie ist direkt als Programmiersprache verwendbar - man spricht in diesem Fall von Logikprogrammierung. Wir führen die prädikatenlogisch basierte Wissensrepräsentation gleich im Kontext von Logikprogrammierung ein.

Logikprogrammierung ist ein direktes Ergebnis von früheren Arbeiten zum automatischen Theorembeweisen und beruht im wesentlichen auf dem Resolutionsprinzip von Robinson (1965). Das Resolutionsprinzip ist ein grundlegendes Verfahren zum mechanischen Deduzieren logischer Formeln aus Axiomen. Die Idee, Logik unmittelbar als Programmiersprache zu benutzen, geht in ihren theoretischen Grundlagen vor allem auf die Arbeiten von Kowalski (vgl. z.B. Kowalski 1988) und Colmerauer zurück (vgl. Lloyd 1993).

Vom Hauptstrom der Computersprachen weicht Logikprogrammierung insofern ab, als - zumindest in ihrer reinen Form - angestrebt wird, *Probleme* lediglich in Form logischer Axiome deklarativ zu *beschreiben*. Eine solche Axiomenmenge bildet eine Alternative zum konventionellen Programm, in dem *Algorithmen* zur Problemlösung zu programmieren sind. Das Programm kann ausgeführt werden, wenn es mit einem Problem - als zu beweisender Aussage formuliert - versorgt wird. Die Ausführung ist ein Versuch, das Problem zu lösen, das heißt, die zu beweisende Aussage mit den Annahmen im Logikprogramm herzuleiten (Sterling/Shapiro 1988: xxiii).

Für die Verwendung der Prädikatenlogik auf Computern ist eine syntaktische Variante der Logik erster Stufe relevant, die Hornclausenlogik, die wiederum ein Spezialfall der Clausenlogik ist. Die verbreitetste Logik-Programmiersprache, PROLOG (PROgramming in LOGic), kann als informatische Realisierung der Hornclausenlogik betrachtet werden. Für das weitere Vorgehen erscheint es zweckmäßig - nach einer kurzen Vorstellung der Clausenlogik - Aufbau und Arbeitsweise von PROLOG zumindest in den Grundzügen darzulegen. Wir knüpfen dabei bereits an das im zweiten Teil behandelte Grundmodell der Balancetheorie von Heider an. Vereinfachungen und Kürzungen werden bewußt in Kauf genommen, da der Rahmen der Arbeit eine detailliertere Darstellung nicht erlaubt.

---

[36] Der im Text schon mehrfach verwendete Zusatz "erster Ordnung" oder "erster Stufe" bedeutet, daß All- und Existenzaussagen nur über Individuen getroffen werden können (für alle x: p(x)), während in der Prädikatenlogik zweiter Stufe auch über Mengen und Eigenschaften quantifiziert werden kann (für alle p: p(x)). Die ausdrucksstärkere Logik zweiter Stufe hat die unangenehme Eigenschaft, daß sie weder vollständig noch entscheidbar ist, während die Logik erster Stufe auch nicht entscheidbar, aber immerhin vollständig ist. Logiken zweiter (und höherer) Ordnung spielen in der KI und Wissenschaftstheorie eine untergeordnete Rolle.

4.2.2.1 Clausen, Hornclausen, PROLOG

Der grundlegende Term bei automatischen Beweisern ist der Begriff der Clause. Eine *Clause* ist eine spezielle prädikatenlogische Formel der folgenden Form:[37]

$B_1 \vee B_2 \vee ... \vee B_m \leftarrow A_1 \wedge A_2 \wedge ... \wedge A_n \quad (m,n \geq 0)$.

In dieser Clause ist "$\vee$" das logische "oder" und "$\wedge$" das logische "und". Das logische "wenn-dann" "$\rightarrow$" wird hier aus Gründen, die mit PROLOG zusammenhängen, umgekehrt geschrieben; "$\leftarrow$" ist somit zu lesen als "dann-wenn".

Atomformeln B_i, A_j (i=1,...,m; j=1,...,n) haben die Form $p(t_1,...,t_k)$, wobei p ein k-stelliges *Prädikat* ist und $t_1,...,t_k$ Terme sind. Ein *Term* ist entweder eine *Variable* $x_1,...,x_k$, eine *Konstante* $a_1,...,a_k$ oder ein Ausdruck der Form $f(t_1,...t_k)$, wobei f ein k-stelliges Funktionssymbol (*Funktor*) und $t_1,...,t_k$ Terme sind. Ein Prädikat p bzw. Funktor f der Stelligkeit k wird auch mit p/k bzw. f/k bezeichnet.

Eine *Hornclause* (benannt nach dem Logiker Horn) ist eine spezielle Clause mit $m \leq 1$. Wir beschränken uns im folgenden auf Hornclausen, also auf Clausen der folgenden Form:

$B_1 \leftarrow A_1 \wedge A_2 \wedge ... \wedge A_n$.

$A_1, A_2,...,A_n$ sind die konjunktiv verknüpften Bedingungen, B_1 ist die Konklusion der Hornclause. Die Konklusion B_1 ist der *Kopf* der Formel, der durch "$\leftarrow$" vom *Rumpf* (Körper) getrennt wird. Die Formel kann gelesen werden als:

B_1, wenn A_1 und A_2 und ... und A_n.

Enthält eine Hornclause Variablen $x_1,..., x_k$, so ist dies zu interpretieren als:

Für alle $x_1,...,x_k$ gilt: $B_1 \leftarrow A_1 \wedge A_2 \wedge ... \wedge A_n$.

Für die folgenden Beispiele sei der Zeichenvorrat der Hornclausenlogik nach PROLOG-Konvention festgelegt (vgl. Kleine-Büning/Schmitgen 1986):

(1) Die Menge der Prädikatsymbole P, Funktionssymbole F und Konstanten K bestehe aus Zeichenketten des lateinischen Alphabets, beginnend mit Kleinbuchstaben;

(2) Variablen V seien: $x_1,...,x_k$;

(3) $P \cap F \cap K \cap V = \varnothing$.

Der so festgelegte Zeichenvorrat entspricht bereits dem der Programmiersprache PROLOG mit Ausnahme von (2): in PROLOG sind Variablen Zeichenketten, die mit Großbuchstaben beginnen.

Im Hinblick auf PROLOG sind vier Fälle zu unterscheiden (vgl. Belli 1986: 60ff.):

(F1) $m = 1, n = 0$: $B_1 \leftarrow$

Dies ist eine (Atom-)Formel, die von keiner Bedingung abhängt; wir schreiben einfachheitshalber: B_1.

Wenn wir im folgenden Prädikate und Definitionen aus der Heider-Theorie verwenden (vgl. S.118ff.), dann wäre z.B. `positiv(a,b)` eine atomare (Horn-) Clause mit dem 2-stelligen Prädikat `positiv/2`, den Konstanten `a` und `b` und der Heider-Interpretation, nach der zwischen a und b eine positive Relation besteht. Ebenso ist `positiv(ehemann(inge_meier),cdu)` eine Atomformel mit der Variation, daß das erste Argument von `positiv/2` ein 1-stelliger Term ist mit dem Funktor `ehemann/1` und dem Argument `inge_meier`.

[37] Vgl. zum folgenden Kowalski (1988), Kap.1.

(F2) $m,n \neq 0$: $B_1 \leftarrow A_1 \wedge A_2 \wedge ... \wedge A_n$

Dies ist der übliche dann-wenn-Fall.

Beispielsweise ist

`gleichgewicht(triade(`x_1`,`x_2`,`x_3`)) ← positiv(`x_1`,`x_2`) ∧`
`negativ(`x_1`,`x_3`) ∧ negativ(`x_2`,`x_3`)`

eine Hornclause mit einer Konklusion und drei Bedingungen. Die Formel soll die Heider-Definition ausdrücken:

Für alle x_1, x_2, x_3: Die Triade x_1, x_2 und x_3 ist im Gleichgewicht, wenn zwischen x_1 und x_2 eine positive Relation, zwischen x_1 und x_3 eine negative Relation und zwischen x_2 und x_3 ebenfalls eine negative Relation besteht. Eine andere Hornclause, die besagt, daß alle Entitäten, die im Gleichgewicht sind, stabil sind, wäre:

`stabil(`x_1`) ← gleichgewicht(`x_1`).`

Variablen gleichen Namens in Kopf und Rumpf sind aneinander gebunden. Formeln des Typs (F2) bezeichnet man als Regeln. Damit ist die Brücke geschlagen zu dem letzten Kapitel, in dem Regeln als allgemeines Repräsentationsschema behandelt wurden, nicht aber deren konkrete Codierung: *Hornclausenlogik erlaubt also eine logische Darstellung von Regeln.*

(F3) $m = 0, n \neq 0$: $\leftarrow A_1 \wedge A_2 \wedge ... \wedge A_n$

Dies ist eine Formel, die nur aus Bedingungen besteht. Sie kann interpretiert werden als:

es ist nicht der Fall, daß gilt: A_1 und A_2 und ... und A_n.

Beispielsweise ist

`← positiv(a,b) ∧ negativ(a,c)`

zu lesen als: "es ist nicht der Fall, daß zwischen a,b eine Positiv- und zwischen a,c eine Negativrelation existiert".

(F4) $n,m = 0$: $\leftarrow$

Dies ist die leere Clause, die in einem PROLOG-Programm selbst nie erscheint, die aber beweistechnisch wichtig ist (siehe S.90-91).

Wir haben uns bislang immer noch im Rahmen der Hornclausenlogik bewegt. Der Schritt von der Hornclausenlogik zu ihrer informatischen Realisierung PROLOG ist aber nur mit minimalen Änderungen verbunden und lediglich bedingt durch die maschinelle Implementierung.[38]

Zunächst wird in PROLOG syntaktisch nicht unterschieden zwischen Funktoren und Prädikaten. Prädikate werden deshalb in der Literatur ebenfalls als Funktoren bezeichnet. Dies hat die Konsequenz, daß die syntaktische Elementarform von PROLOG nicht die Atomformel ist, sondern der Term: Terme sind allgemeiner als Atomformeln, da jede Verknüpfung von Termen wiederum Terme ergibt. Die rekursive Definition ermöglicht, daß PROLOG den Term als einzige syntaktische Grundform kennt.

---

[38] An dieser Stelle sei jedoch schon darauf hingewiesen, daß real existierende PROLOG-Interpreter Sprachkonstrukte enthalten, welche die Hornclausenlogik erweitern bzw. durchbrechen (vgl. die Hinweise S.91-92).

Zusammengesetzte Terme heißen in PROLOG auch Strukturen und lassen sich als Bäume darstellen, wie etwa `g(f(X),a,h(Y,X,b))`.

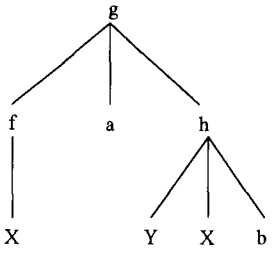


Abb. 4.2: Der PROLOG-Term `g(f(X),a,h(Y,X,b))` als Baumstruktur

PROLOG-Terme sind Variablen, Konstanten oder Strukturen. Variablen sind - im Unterschied zur Clausenlogik - Zeichenketten beginnend mit Großbuchstaben. Einen Sonderfall bildet die sog. anonyme Variable "_", die immer dann benutzt wird, wenn es auf den Wert der Variable nicht ankommt. Konstanten sind Integers oder Zeichenketten, die mit Kleinbuchstaben beginnen oder in Hochkomma eingeschlossen sind. Konstanten können im übrigen als Funktoren der Stelligkeit 0 betrachtet werden.

Wir können nun PROLOG-Programme konkreter charakterisieren. Ein PROLOG-Programm ist eine Menge von Hornclausen, die unterschieden werden in

- Fakten,
- Regeln,
- Fragen.

Das eigentliche Programm besteht nur aus Fakten und Regeln. Fakten entsprechen variablenfreien Hornclausen der Gestalt (F1), Regeln Hornclausen der Gestalt (F2) und Anfragen solchen der Gestalt (F3). Die oben gegebenen Heider-Beispiele sehen in der verbreitetsten PROLOG-Notation (Clocksin/Mellish 1987) wie folgt aus.

```
positiv(a,b).
positiv(ehemann(inge_meier),cdu).
gleichgewicht(triade(X,Y,Z)) :-
   positiv(X,Y),
   negativ(X,Z),
   negativ(Y,Z).
stabil(X) :- gleichgewicht(X).

?- positiv(a,b), negativ(a,c).
```

Im Vergleich zur Hornclausenlogik sind folgende Notationsänderungen zu verzeichnen:

- bei Regeln statt "←" ":-" ;
- statt "∧" ein Komma;
- bei Anfragen statt "←" ein "?-";
- Variablen sind Strings, beginnend mit Großbuchstaben;
- jede Clause ist mit einem Punkt abzuschließen.

Wie läßt sich nun die Hornclausenlogik bzw. PROLOG zur Modellierung verwenden? Im allgemeinen eignen sich Fakten zur Darstellung von Eigenschaften von Objekten und Beziehungen zwischen Objekten. Regeln können zur Repräsentation von Gesetzen (Allaussagen) und zur Definition neuer, komplexerer Relationen auf der Basis existierender Beziehungen verwendet werden. Anfragen schließlich dienen zur Gewinnung von Informationen aus dem Programm. Aus dem eben Dargelegten ergeben sich etwas systematischer folgende Modellierungsmöglichkeiten mit Logik und PROLOG (für weitere Details vgl. Reimer 1991: 35-54):
Eigenschaften lassen sich mit 1-stelligen Prädikaten repräsentieren, z.B. "a ist Philosoph" mit `philosoph(a).` n-stellige Beziehungen können durch n-stellige Prädikate ausgedrückt werden, z.B. "a ist verheiratet mit b" durch

```
verheiratet_mit(a,b).
```

oder "a sanktioniert b mit c" durch

```
sanktioniert_mit(a,b,c).
```

Einfache zeitliche Aspekte können durch Einführung einer zusätzlichen Argumentstelle dargestellt werden, z.B. "a ist mit b verheiratet zum Zeitpunkt 2" durch `verheiratet_mit(a,b,2).` Unsicheres Wissen kann durch Zuordnung numerischer Sicherheitsfaktoren ebenfalls durch eine weitere Argumentstelle abgebildet werden, z.B. "a ist ein ziemlich guter Freund von b" durch: `freund(a,b,0.8).` Die Zahl 0.8 an dritter Argumentstelle soll dabei ausdrücken, daß die Aussage "ziemlich sicher" zutrifft.

Definitionen und Allaussagen wie "alle Entitäten mit der Eigenschaft P haben die Eigenschaft Q" können mit Regeln ausgedrückt werden. Der Allsatz "Alle Soziologen haben eine Statistikausbildung" kann formuliert werden als:

```
hat_statistik_ausbildung(X) :- soziologe(X).
```

Das Analoge gilt für Definitionen. "x ist Mutter von y, wenn y Kind von x ist und x weiblich ist" läßt sich mit folgender PROLOG-Regel repräsentieren:

```
mutter(X,Y) :- kind(Y,X), weiblich(X).
```

Formale relationale Eigenschaften wie Symmetrie von Prädikaten können ebenfalls in Regelform ausgedrückt werden, z.B. die Symmetrie der Heiratsrelation durch:

```
verheiratet_mit(X,Y) :- verheiratet_mit(Y,X).
```

Angemerkt sei, daß eine PROLOG-Regel *immer* die gleiche syntaktische Grundstruktur hat, nämlich: *genau eine* Atomformel im Kopf (Konklusion) und *eine oder mehrere* Atomformeln im Rumpf (Bedingungen). Die Atomformeln im Rumpf können durch "und" (in PROLOG: Komma) oder "oder" (in PROLOG: Semikolon) logisch verknüpft sein.

Die genaue Syntax (und Semantik) von PROLOG ist beschrieben in Bratko (1987: 29-67), Clocksin/Mellish (1987: 21-42) oder Kleine-Büning/Schmitgen (1986: 70-78).

4.2.2.2 Deduktion in PROLOG

Bislang wurde nur die *Repräsentation* von Wissen mit PROLOG dargestellt, nun soll die *Wissensmanipulation*, also die Herleitung von neuem Wissen, betrachtet werden. Die Deduktion neuen Wissens kann in PROLOG als Theorembeweisen angesehen werden: PROLOG kann als automatischer Theorembeweiser interpretiert werden, der

aus Axiomen Theoreme herleitet. Das Deduktionsprinzip von PROLOG soll nun konkret an einem einfachen Beispiel gezeigt werden.

Wir benutzen für die weiteren Erläuterungen die folgende einfache Wissensbasis mit der obengenannten Interpretation, bestehend aus sechs Fakten und zwei Regeln:

```
positiv(a,b).
positiv(a,d).
negativ(a,f).
negativ(b,c).
negativ(d,c).
negativ(a,c).

gleichgewicht(triade(X,Y,Z)) :-
   positiv(X,Y),
   negativ(X,Z),
   negativ(Y,Z).

stabil(X) :-
   gleichgewicht(X).
```

Eine Anfrage an eine gegebene Wissensbasis wird als *Beweisziel* oder *Goal* interpretiert.

Das Beweisziel

```
?- positiv(a,b), positiv(a,c).
```

besteht aus zwei konjunktiv verknüpften Termen, also zwei zu beweisenden Teilzielen. Das erste Goal kann bewiesen werden, da es mit dem entsprechenden Fakt in der Datenbasis "matcht". Ein Term *matcht*[39] (oder *unifiziert*) mit einem anderen Term, falls

- beide Terme gleich sind - was hier der Fall ist -, oder
- falls Variablen im Spiel sind, diese in beiden Termen so gebunden werden können, daß nach der Substitution der Variablen beide Terme gleich sind (Bratko 1987: 38).

Die Substituierung einer Variablen durch einen Term (Konstante, Struktur oder andere Variable) nennt man *Instantiierung*, und die Stelle in der Datenbasis wird markiert.

Das zweite Teilziel `positiv(a,c)` ist nicht erfolgreich und scheitert, da kein entsprechender Term in der Datenbasis matcht. Mit dem Scheitern des zweiten Teilziels scheitert das ganze Beweisziel, und es wird die Antwort `no` ausgegeben. PROLOG antwortet bei Anfragen *ohne* Variablen grundsätzlich mit `yes` oder `no` . Im ersten Fall bedeutet dies einen *erfolgreichen* Beweis, d.h. daß der angefragte Fakt logisch aus der bestehenden Wissensbasis deduziert werden kann, im zweiten Fall bedeutet dies ein Scheitern, d.h. der Fakt folgt nicht aus der aktuell bestehenden Wissensbasis.

Bei Abfragen *mit* Variablen wird die Instantiierung der Variablen ausgegeben.

Die Anfrage

```
?- positiv(a,X).
```

matcht als erstes mit dem Fakt `positiv(a,b)` und liefert als Output

```
X = b.
```

[39] Von "Pattern-Matching" (Mustererkennung); "matchen" könnte man vielleicht am besten mit "passen" übersetzen, wir behalten aber, wie in der Literatur üblich, den Originalterm bei.

Die Variable X wird hierbei gebunden an b bzw. unifiziert mit b.

Ein anschließend eingegebenes Semikolon ; mit der Bedeutung des logischen "oder" hebt die Variablenbindung wieder auf, und PROLOG sucht die nächste Lösung:

```
X = d.
```

Eine weitere Alternativlösung existiert nicht, und PROLOG würde auf eine entsprechende Aufforderung mit no antworten.

Logisch betrachtet sind Variablen in Fragen existenzquantifiziert (Sterling/Shapiro 1988: 6). Die PROLOG-Frage

```
?- positiv(a,X)
```

ist also zu lesen als: Gibt es ein X, so daß `positiv(a,X)` gilt; die konjunktive Anfrage

```
?- p(X), q(X)
```

als: Gibt es ein solches X, daß beides wahr ist, `p(X)` und `q(X)`.[40]

Wir haben uns bislang bei der Erläuterung des Beweisverfahrens auf Fakten beschränkt. Für die Anfrage

```
?- gleichgewicht(Triade).
```

ist kein Fakt vorhanden, aber ein Prädikat gleicher Stelligkeit findet sich als Konklusion einer Regel. Die Frage matcht also mit dem Kopf der obigen Regel, wobei die Variable `Triade` in der Regel instantiiert wird mit dem Term `triade(X,Y,Z)`. Grundsätzlich sind Variablen in Regeln allquantifiziert.

PROLOG geht nun als *rückwärtsverkettender Regelinterpreter* vor: es versucht nach und nach, alle Bedingungen (Teilziele) der Regel mit den entsprechenden Instantiierungen zu beweisen. *Dieser Regelinterpreter ist impliziter Teil der Sprache PROLOG und muß nicht extra erstellt werden.* Das erste Teilziel `positiv(X,Y)` matcht mit `positiv(a,b)`, so daß X nun gebunden wird an a und Y an b. Wird eine Variable innerhalb einer Regel gebunden, so ist sie bei allen Vorkommnissen in der Regel an denselben Wert gebunden. Das zweite Teilziel lautet also folglich `negativ(a,Z)`. `negativ(a,Z)` matcht mit `negativ(a,f)`, wobei Z=f. Das dritte Teilziel heißt somit jetzt `negativ(b,f)`. Eine Inspektion der Faktenbasis zeigt, daß kein entsprechender Eintrag vorhanden ist, und der Beweis für das dritte Teilziel scheitert damit. Nun setzt der für die Lösungssuche zentrale Mechanismus des *Backtracking* (Rücksetzen) ein: da das dritte Beweisziel gescheitert ist, erfolgt eine Lösung der gebundenen Variablen Z=f und ein Rücksetzen zum zweiten Teilziel. Es wird nun ein erneuter, alternativer Beweis für `negativ(a,Z)` versucht. Mit `negativ(a,Z)` matcht jetzt der letzte Fakteneintrag `negativ(a,c)`, wobei Z=c. Das dritte Teilziel lautet also `negativ(b,c)` und ist als Eintrag in der Datenbasis ebenfalls vorhanden. Somit sind sämtliche Teilziele der Regel mit den dabei erfolgten Instantiierungen bewiesen, und es wird die Antwort ausgegeben:

```
Triade = triade(a,b,c).
```

Ein nun eingegebenes Semikolon würde ein vom Benutzer initiiertes Backtracking auslösen. PROLOG würde nach einer zweiten Lösung suchen und diese mit

```
Triade = triade(a,d,c)
```

---

[40] Variablen in Fakten sind ebenfalls allquantifiziert. Allerdings kommen in der Praxis Variablen in Fakten nicht vor.

finden. Eine weitere Alternativ-Lösung existiert nicht und die Eingabe von ; würde mit `no` beantwortet werden.

Wir beschließen damit die elementare Einführung in den PROLOG-Beweismechanismus und fassen die grundlegende Arbeitsweise von PROLOG zusammen: PROLOG-Ableitungen erfolgen über (Pattern-)Matching, Top-Down-Suche, Backward-Chaining und Backtracking:

- Matching: Für jedes Beweisziel wird ein Matching versucht. Ein Term T1 matcht mit einem Term T2, wenn Funktor und Stelligkeit von T1 und T2 identisch sind, sowie die Argumente entweder gleich sind oder durch Variablenbindung (Instantiierung) gleich gemacht werden können.
- Top-Down: Für jedes Beweisziel wird in der Faktenbasis *von oben nach unten* das Matching versucht.
- Backward-Chaining: ist das Beweisziel der Kopf einer Regel (die Konklusion), so wird ausgehend von der Konklusion versucht, nach und nach die Bedingungen zu beweisen. Ist ein Teilziel wiederum ein Regelkopf, wird erst diese Regel - die eventuell wiederum weitere Regelköpfe als Teilziele enthalten kann - abgearbeitet, bevor die weiteren Teilziele geprüft werden. Damit ist PROLOG ein *rückwärtsverkettender Regelinterpreter*.
- Backtracking: Führt ein Teilziel nicht zum Erfolg, werden die Instantiierungen, die bei diesem Teilziel erfolgten, gelöst, und es erfolgt ein Zurücksetzen (Backtracking) zum vorhergehenden Teilziel. Die Eingabe eines Semikolons nach Ausgabe einer Lösung bewirkt ein vom Benutzer initiiertes Backtracking.

Ein PROLOG-Interpreter implementiert somit unmittelbar eine rückwärtsverarbeitende Top-Down Inferenzmaschine, ohne daß dies explizit programmiert werden muß. Angemerkt sei noch, daß logikbasierten Repräsentationen und insbesondere PROLOG die "Closed World Assumption" zugrunde liegt. Diese Annahme besagt, daß eine Aussage als nicht wahr angesehen wird, wenn sie aus der *vorliegenden* Wissensbasis nicht abgeleitet werden kann.

4.2.2.3 Listen, Mengen und Rekursion

Bei der Repräsentation von Theorien spielen Mengen eine wichtige Rolle. Mengen werden in PROLOG in Listen repräsentiert.

Eine Liste

- ist entweder leer,
- oder besteht aus mehreren Elementen.

Die leere Liste wird dargestellt als `[]`, eine nicht-leere Liste durch `[e1,e2,e3,....,en]`. Das erste Element einer nicht-leeren Liste - `e1` - ist der *Kopf*, der Rest `[e2,e3,...,en]` ist der *Rumpf* (*Rest*, *Schwanz*) der Liste. Der Rumpf einer Liste ist wieder ein Liste, der Kopf dagegen nicht. Der Rumpf hat abermals einen Kopf `e2` und den Rumpf `[e3,...,en]` usw. Die Liste `[en]` schließlich hat den Kopf `en` und den Rumpf `[]`.

Alle Elemente einer Liste müssen vom Typ Term sein. Die Liste selbst ist auch ein Term (genauer gesagt: eine Struktur), d.h. die Elemente einer Liste können wiederum Listen sein, deren Elemente wiederum Listen sein können etc.

Beispiele für gültige Listen sind:

```
[a,b,c]
[a,1,a,b,2,d]
[mo,di,mi,do,fr,sa]
['Beispiel',einer,liste,in,'Prolog']
[dies, ist, [eine,[verschachtelte,liste]]]
```

Da in Listen Elemente mehrfach vorkommen dürfen und der Reihenfolge nach geordnet sind, sind Listen streng genommen keine Mengen. Trotzdem können Listen natürlich zur Repräsentation von Mengen verwendet werden.

Für die Trennung einer Liste in Kopf und Rumpf steht der Listen-Separator | zur Verfügung: mit `[X|Y]` wird `X` instantiiert auf den Kopf der Liste und `Y` auf den Rumpf. In der folgenden Tabelle finden sich einige Beispiele für Unifikation/Matching zweier Listen.

Tab. 4.1: Drei Instantiierungsbeispiele

Liste1	Liste2	Instantiierung
`[a]`	`[X|Y]`	`X=a Y=[]`
`[a,b,c]`	`[X|Y]`	`X=a Y=[b,c]`
`[1,2,3]`	`[X,Y|Z]`	`X=1 Y=2 Z=[3]`

In Zusammenhang mit Listen und Listenoperationen - aber nicht nur damit - sind Rekursion und rekursive Definitionen von zentraler Bedeutung. Eine rekursiv definierte Regel ist eine Regel, die sich selbst wieder aufruft, d.h. der Kopf der Regel erscheint im Regelkörper noch einmal. Ein beliebtes Beispiel ist die `member`-Relation, die prüft, ob ein Element in einer Liste enthalten ist. Beispielsweise liefert

```
?- member(a,[b,c,a]).
yes
```

weil `a` Element der Liste `[b,c,a]` ist,

```
?- member(a,[b,d]).
no
```

weil `a` nicht in der Liste enthalten ist.
Das Programm für die `member`-Relation kann rekursiv wie folgt definiert werden:
X ist Mitglied einer Liste L, wenn
(1) X der Kopf von L ist oder
(2) X ein Mitglied des Rests von L ist.
Dies kann in Form von zwei PROLOG-Clausen beschrieben werden:

```
member(X,[X|_]).
member(X,[_|Rest]) :-
   member(X,Rest).
```

`member(X,[X|_])` ist dabei der Rekursionsanfang, der als Faktum vor der Regel steht. Eine Anfrage ist erfolgreich, wenn `X` Kopf der Liste ist. Andernfalls scheitert die

erste Clause, und es wird ein Beweis mit der zweiten Clause - einer Regel - versucht. Die Regel ruft sich rekursiv selbst wieder auf mit der Rumpfliste an zweiter Argumentstelle, und die erste Clause prüft nun wieder, ob `X` Kopf der (Rumpf-)Liste ist usw.

Generell gilt, daß auf Listenelemente nur über den Kopf zugegriffen werden kann. Um ein Element mit einer bestimmten Eigenschaft zu finden, muß also zunächst der Kopf geprüft werden. Ist die Prüfung negativ, so wird die Liste "geköpft", d.h. das erste Element wird entfernt, und der Kopf des Rumpfes wird geprüft. Der Prüfprozeß wird solange fortgesetzt, bis das gesuchte Element gefunden ist oder die leere Liste übrig bleibt.

Nicht alle Argumente von `member/2` müssen gebundene Variablen sein. Enthält die erste Argumentstelle eine Variable, die zweite eine variablenfreie Liste, so erhalten wir als erste Lösung eine Unifikation der Variablen mit dem ersten Listenelement, als zweite eine Unifikation mit dem zweiten usw.

```
member(X,[a,b,c,d]).
```

würde also liefern

```
X = a
```

und ein anschließend ausgelöstes Backtracking mit ;

```
X = b
```

usw.

Man sieht daran, daß einfache Prädikate wie `member/2` flexibel genutzt werden können.

Als abschließendes Beispiel definieren wir ein Prädikat, das die Gleichheit zweier Mengen prüft und später benötigt wird. Da Listen in PROLOG nur gleich sind, wenn diese die gleichen Elemente in übereinstimmender Reihenfolge enthalten, sind die Listen `[a,b,c,d]` und `[b,a,c,d]` nicht identisch in PROLOG, obwohl sie mengentheoretisch gleich sind. Eine einfache Gleichheitsprüfung von Listen ist für Mengen also nicht geeignet.

Wir müssen deshalb ein entsprechendes Prädikat definieren: das Prädikat `gleiche_menge(X,Y)` soll wahr sein, wenn `X` und `Y` mengentheoretisch gleich sind. Wir setzen hier voraus, daß weder `X` noch `Y` doppelte Vorkommnisse von Elementen enthalten, also echte Mengen sind - was wiederum mit einem eigens definierten Prädikat geprüft werden könnte.

Wir definieren das Prädikat `gleiche_menge(X,Y)` unter Ausnutzung der mengentheoretischen Grundbeziehung:

$X = Y$ gdw $X \subseteq Y$ und $Y \subseteq X$.

Daß die Listen X und Y (mengentheoretisch) gleich sind, läßt sich damit einfach unter Reduktion auf die Teilmengen-Relation definieren:

```
gleiche_menge(X,Y) :-
   teilmenge(X,Y),
   teilmenge(Y,X).
```

Die rekursive Definition der Teilmengen-Relation lautet:

(1) Die leere Menge ist Teilmenge jeder Menge.
(2) X ist Teilmenge von Y, wenn der Kopf von X Mitglied von Y ist und der Rumpf Teilmenge von Y ist.

```
teilmenge([],Y).
teilmenge([K|X],Y) :-
   member(K,Y),
   teilmenge(X,Y).
```

Das letzte Beispiel eignet sich zur Exemplifizierung von drei wichtigen Eigenschaften von PROLOG, die es für die Modellierung so interessant machen:

1. Die semantische Lücke zwischen Quellsprache (hier: Mengenlehre) und PROLOG als Zielsprache ist gering. Die Definition der mengentheoretischen Gleichheit in PROLOG kann - zumindest auf der oberen Ebene - genauso geschrieben werden wie in der Quellsprache.
2. Es muß kein Algorithmus festgelegt werden, der bestimmt, welche Schritte zur Verifizierung der Mengengleichheit ausgeführt werden müssen. Vielmehr erfolgt die Definition der Mengengleichheit rein beschreibend.
3. Komplexe Prädikate lassen sich auf weniger komplexe, immer einfachere Prädikate zurückführen, so daß sich die oben erwähnte hierarchische Strukturierung ergibt: das Prädikat `gleiche_menge/2` wird unter Rückgriff auf `teilmenge/2` definiert und dieses unter Rückgriff auf `member/2`. Wie im nächsten Kapitel gezeigt wird, können Prädikate auch als Prozeduren interpretiert werden, so daß das Beispiel den oben eingeführten Begriff der Prozedurabstraktion verdeutlicht.

Mit PROLOG-Listen sind alle gebräuchlichen mengensprachlichen Operationen durchführbar. Typische Operationen auf Mengen sind Durchschnitts-, Vereinigungs-, Differenzmengenbildung oder die Bildung des Kreuzprodukts zweier Mengen. Das Kreuzprodukt würde eine Menge von geordneten Paaren liefern, in PROLOG eine Liste von zweielementigen Listen: `[[a,b],[a,c],[a,d],...]`. Die Elemente der inneren Listen (also die der Listenpaare) sind hier als geordnet aufzufassen und die Elemente der äußeren Liste als ungeordnet. Soll dieser Unterschied deutlicher hervorgehoben werden, kann die Kreuzproduktliste auch so dargestellt werden:

```
[(a,b),(a,c),(a,d),...].
```

Allgemeine, nicht ausschließlich mengenspezifische Listenoperationen sind z.B.: Listen zusammenhängen, Listenelemente entfernen, Bestimmung des letzten Listenelements, Länge einer Liste bestimmen, Listenpermutationen, Listenumkehr oder Sortieren von Listen. Prädikate zur Definition dieser Operationen finden sich z.B. in Kleine-Büning/Schmitgen (1986) oder Belli (1986).

4.2.2.4 Interpretation und Beweisverfahren von PROLOG-Programmen

Wir haben bislang die deklarative Sicht der Logikprogrammierung eingenommen und werden diese im folgenden auch weitgehend beibehalten. In der deklarativen oder logischen Bedeutung *beschreibt* ein Logikprogramm einen Wissensausschnitt mit Fakten und Regeln. Fakten und Regeln werden als eine Menge von Axiomen verstanden, die Anfrage als ein vermutetes Theorem und PROLOG versucht, dieses Theorem

zu beweisen und aus den Axiomen abzuleiten (Bratko 1987: 21). Die deklarative Bedeutung bestimmt also, ob ein Ziel wahr ist, und falls ja, für welche Variablenwerte. Die Regel

```
p(X) :- q(X), r(X)
```

ist damit unter logischer Betrachtungsweise zu lesen als:

```
p(X), wenn q(X) und r(X).
```

oder

`p(X)` ist wahr, wenn `q(X)` wahr ist und `r(X)` wahr ist.

Die Regeln können dabei als Axiome betrachtet werden, die zur Definition von *neuen und komplexeren Relationen* unter Verwendung von bereits *bestehenden, einfachen Relationen* verwendet werden.

Soweit möglich, sollte PROLOG deklarativ gesehen werden, und es sollte insbesondere die bevorzugte Interpretation des Modellbauers sein. Obwohl die deklarative Betrachtungsweise elegant, einfach und vorteilhaft ist, ist in der Praxis aber oft die prozedurale Betrachtung nicht zu vermeiden.[41] Die prozedurale Bedeutung von PROLOG-Programmen bestimmt, *wie* eine Ausgabe erhalten wird, d.h. wie das PROLOG-System Fakten und Regeln auswertet. Prozedural interpretiert ist eine Anfrage eine "ereignisgesteuerte Aktivierung einer Prozedur" (Schnupp/Nguyen Huu 1987: 60): Anfragen lösen einen Prozeduraufruf bzw. eine Sequenz von Prozeduraufrufen aus, wobei der Funktor eines Terms den *Prozedurnamen* darstellt und die Argumente die *Übergabeparameter*. Eine Frage löst dann den Aufruf eines Faktes oder des Kopfes einer Regel aus. Im Fall des Aufrufes einer Regel heißt die Ausführung der Prozedur, alle Prozeduren im Regelrumpf der Reihenfolge nach auszuführen.

Die obengenannte Regel ist prozedural damit wie folgt zu lesen:

Um die Prozedur `p(X)` zu lösen, rufe die (Unter-)Prozedur `q(X)` auf und dann die (Unter)Prozedur `r(X)`, bzw. mit dem Aufruf `p(a)`:

Um die Prozedur `p(a)` zu lösen, rufe `p(X)` auf mit dem Parameter `a` usw.

Die prozedurale Bedeutung ist die Sicht, die der PROLOG-Interpreter einnimmt. Sie bestimmt, wie - nach welchem Suchalgorithmus - der implizite Inferenzmechanismus arbeitet. Wir haben oben den grundlegenden Ableitungsvorgang dargestellt und PROLOG als *implizites Backward-Chaining-System* vorgestellt, ohne für beides eine Begründung gegeben zu haben. Wir holen die Begründung hier nach. Das Ableitungsverfahren und damit die prozedurale Interpretation von PROLOG beruht auf dem Resolutionsalgorithmus, einem Ableitungsprinzip, das zum mechanischen Beweisen von Theoremen benutzt werden kann.

Das *Resolutionsprinzip* (Robinson 1965) besagt folgendes: Zwei Clausen können resolviert werden, wenn zu einer Atomformel der einen Clause die gleiche Atomformel in der anderen Clause *negiert* vorkommt. Die Vereinigung der beiden alten Clausen ohne das komplementäre Paar von Atomformeln erzeugt eine neue Clause, die Resolvente.

Zum Beispiel resolvieren die beiden Clausen

---

[41] Manche Autoren (z.B. Schnupp 1987) interpretieren PROLOG fast nur prozedural. Diese Betrachtungsweise ist typisch für Programmierer, die aus den prozeduralen/algorithmischen Sprachen FORTRAN, PASCAL, C o.ä. kommen.

p und
$(q \vee \neg p)$
zur Resolvente q, was dem Modus Ponens entspricht.

Ein wichtiger Satz lautet, daß eine Clausenmenge M genau dann widerspruchsvoll ist, wenn die Resolution auf M zur leeren Clause (F4) führt (vgl. Kleine-Büning/Schmitgen 1986). Dieser Satz wird zur maschinellen Herleitung benutzt. Der PROLOG-Inferenzmechanismus führt - logisch betrachtet - einen Beweis durch Widerspruch: (F3) besagt nämlich, daß PROLOG-Anfragen als negierte Fakten zu verstehen sind. Die zu beweisende Anfrage wird als negiertes Theorem nun solange resolviert, bis die leere Clause (F4) hergeleitet ist. In diesem Fall ist gezeigt, daß die Negierung des Theorems zu einem Widerspruch führt und das Theorem damit gültig ist.

Bei prädikatenlogischen Formeln müssen Variablen bzw. Terme beim Resolvieren durch Variablensubstitution so gleichgesetzt werden, daß ein Widerspruch erzeugt wird - was u.U. sehr aufwendig ist. Die theoretische Basis für das Resolvieren in der Prädikatenlogik ist das Theorem von Herbrandt und Gegenstand der Unifikationstheorie.

Im Rahmen dieser Arbeit ist eine weitergehende Behandlung des Resolutionsprinzips nicht möglich. Eine ausführliche und genauere Darstellung geben Kleine-Büning/Schmitgen (1986), Clocksin/Mellish (1987) und insbesondere Lloyd (1993). Ein zusammenfassender Überblick findet sich in Manhart (1988).

Die Aussage mancher Autoren, nach der PROLOG eine deklarative, nicht-prozedurale Sprache ist, ist also nicht ganz richtig (z.B. Brent 1986: 275). Vielmehr kann PROLOG auf die eine oder andere Weise interpretiert und gelesen werden. Das Besondere und Schöne an PROLOG als Modellierungsinstrument ist nun, daß man das algorithmische Denken - so man will und es die Problemstellung zuläßt - vergessen kann:

"PROLOG enthält den Keim einer großartigen Idee. Ein PROLOG-Programmierer legt nicht fest, wie der Computer seine Aufgaben erledigen soll, sondern er gibt vielmehr eine Beschreibung der Aufgabe als Folge von Bedingungen, die erfüllt werden müssen. Kurz gesagt, in LISP muß man das 'Wie' der Datenverarbeitung spezifizieren, während man in PROLOG nur das 'Was' anzugeben braucht - es ist dann Aufgabe der Maschine, das 'Wie' zu bestimmen. Der letztgenannte Ansatz ist deshalb von Vorteil, weil er den Programmierer davon befreit, sich über die Details der Algorithmen den Kopf zu zerbrechen, mit denen er die Aufgaben zu bewältigen trachtet. Stattdessen braucht er sich nur um eine präzise Aufgabenstellung zu kümmern" (Harmon/King 1986: 103).

Angemerkt sei noch, daß neben dem hier vorgestellten "pure PROLOG" jeder PROLOG-Interpreter Erweiterungen zur Verfügung stellen muß, die mit dem logischen Programmiermodell brechen. Dies sind insbesondere (vgl. GMD 1987):

- *Arithmetik:* arithmetische Prädikate schließen PROLOG an eine maschinennahe Arithmetik an, stehen aber außerhalb des logischen Programmiermodells. Beispielsweise wertet `is/2` arithmetische Ausdrücke aus:

```
?- X is 3+5
X = 8
```

- *Meta-Logik:* Meta-logische Prädikate liegen *überhalb* der gegebenen PROLOG-Sprache und erweitern als Meta-Prädikate die Verwendungsmöglichkeiten von PROLOG. Ein solches Meta-Prädikat ist beispielsweise `var(X)`. Dieses Prädikat ist wahr, wenn `X` eine ungebundene Variable ist. Entsprechend ist das Gegenstück `nonvar(X)` wahr, wenn `X` gebunden ist.
- *Extra-Logik:* Extra-logische Prädikate liegen *außerhalb* des prädikatenlogischen Modells. Ein wesentlicher Typ sind die Ein-/Ausgabe-Prädikate `read/1` und `write/1`.
- *Ablaufsteuerung:* Hierzu zählt vor allem der cut `!/0`, der den Ablauf eines PROLOG-Programms ändert und Backtracking verhindert. Zur Ablaufsteuerung wird manchmal auch das `not/1` gerechnet, da es in den meisten PROLOG-Systemen nicht im logischen Sinn implementiert ist, sondern als "negation by failure".
- *Datenbankzugriffe:* Hierzu zählen vor allem Prädikate zum Hinzufügen (`assert/2`) und Löschen (`retract/2`) von Fakten und Regeln aus der Datenbasis.

Der vorgestellte kleine Ausschnitt von PROLOG muß hier genügen und dürfte hinreichend sein, die Implementierungen der Balancetheorien nachzuvollziehen. Für eine weitergehende Behandlung von PROLOG vgl. Clocksin/Mellish (1987). Eine umfassendere Einführung mit Bezug zur theoretischen Logikprogrammierung und fortgeschrittenen Programmiertechniken ist Sterling/Shapiro (1988). In diesem Buch werden auch theoretische Aspekte ausführlich behandelt, wie z.B. Fragen zu Korrektheit und Vollständigkeit von PROLOG-Programmen. Die mathematischen Grundlagen der Logikprogrammierung finden sich in Lloyd (1993). Praktisch orientierte Lehrbücher sind Belli (1986), Schnupp (1986) und Schnupp/Nguyen Huu (1987). Der im folgenden verwendete PROLOG-Interpreter IF/PROLOG hält sich an den Quasi-Standard Clocksin/Mellish (1987).

4.3 Wissensbasierte Modelle

Nachdem die grundlegenden Elemente regelbasierter Systeme und prädikatenlogischer Wissensdarstellung in PROLOG referiert wurden, ist die Relation dieser beiden Formalismen zu Theorien einerseits und Computermodellen andererseits leicht herzustellen.

4.3.1 Wissensbasierte Systeme und wissenschaftliche Theorien

Ganz allgemein kann der Zusammenhang zwischen Theorien und wissensbasierten Systemen wie folgt gesehen werden: In wissensbasierten Systemen wird das bereichsspezifische Wissen menschlicher Experten repräsentiert. Wissenschaftliche Theorien enthalten ebenfalls bereichsspezifisches Wissen und Wissenschaftler, die mit einer

Theorie arbeiten oder diese gut kennen, können als Experten für diese Theorie betrachtet werden.

Theorien eines bestimmten Wissensgebietes können somit als wissensbasierte Systeme formalisiert werden. Wir sprechen in diesem Fall von *wissens- oder regelbasierten Computermodellen.*

Wissensbasierte Modelle sind mindestens in jenen Bereichen gut anwendbar, in denen konventionelle Simulationstypen nicht brauchbar sind. Sowohl verbale als auch formalisierte strukturelle Theorien sind deshalb zunächst Kandidaten für wissensbasierte Modelle. Inwieweit es Sinn macht, auch quantitative Theorien wissensbasiert zu modellieren, kann hier nicht behandelt werden.

Wenn wir die Wissensrepräsentation auf PROLOG beziehen, dann entsprechen aus der Sicht des Modellierers PROLOG-Fakten den Daten einer Theorie und bereits abgeleiteten Schlußfolgerungen aus der Theorie. Die Daten in Form von PROLOG-Fakten können z.B. Relationen sein oder komplexere Strukturen wie Bäume, Graphen etc. PROLOG-Regeln eignen sich zur Darstellung von Definitionen und Gesetzen. Eine in PROLOG repräsentierte Theorie wäre eine Menge von PROLOG-Regeln, eventuell erweitert um Fakten. PROLOG-Fragen sind Anfragen an die Theorie bezüglich der Gültigkeit von Theoriedaten und Theoremen. Beispiele folgen im zweiten Teil. Die folgende Tabelle listet die wichtigsten Parallelen stichwortartig auf.

Tab. 4.2: Parallelen zwischen (Sozial-)Wissenschaft und Expertensystemen

(Sozial-)Wissenschaft	*Expertensysteme*
Wissenschaftler	Experte
Gesetz	PROLOG-Regel(n)
Definition	PROLOG-Regel(n)
Daten	PROLOG-Fakten
Theorie	Menge von PROLOG-Regeln und Fakten
Frage nach Gültigkeit	PROLOG-Frage
Schlußfolgerung	PROLOG-Inferenz (Resolution)

Betrachten wir die Vorzüge regelbasierter Modelle, so kann ein regelbasiertes Computermodell zunächst natürlich als ganz gewöhnliches Computermodell angesehen werden, so daß die oben genannten Eigenschaften auch auf diese Programme zutreffen. Die Nutzung von Wissensverarbeitung und Expertensystem-Architektur beseitigt jedoch weitgehend die mit den traditionellen Computermodellen verknüpften Nachteile und bringt eine Reihe von gewichtigen, zusätzlichen Vorteilen. Wir fassen diese in sieben Punkten zusammen.

1. *Symbolische, nicht-numerische Repräsentation*
   Ein erster, ganz allgemeiner Vorteil wissensbasierter Modellierung ist die *Loslösung* vom Zwang zum *Quantifizieren* und der Notwendigkeit der *Anwendung numerischer Verfahren.* In qualitativen Theorien sind nicht-numerische Wissensstrukturen enthalten, die in wissensbasierte Systeme bzw. symbolisch orientierte Programmiersprachen besser abgebildet werden können. Dadurch erschließt sich den Sozialwissenschaften ein wesentlich breiteres Anwendungsgebiet für maschinelle Modellierungsformen als mit numerischen Instrumenten.

2. *Weitgehend deklarative, nicht-algorithmische Repräsentation*
   Lindenberg (1971: 85) deutet die Turing-These so, daß, wenn wir bei der Theorienbildung in Algorithmen statt in Propositionen denken, uns wenigstens der Computer zu Hilfe kommt. Dem läßt sich in einem wissensbasierten Verfahren entgegnen: Wenn wir eine deklarative Sprache wie PROLOG verwenden, dann kommt uns der Computer zu Hilfe, *und* wir können weiter in Propositionen denken. Der wissensbasierte Ansatz erlaubt eine weitgehende Abkehr vom *prozedural-algorithmischen Denken* des Modellierers. Zwar muß zwangsläufig *jedem* Computerprogramm ein Algorithmus unterliegen, wie eben dargelegt wurde, ist es aber in PROLOG möglich, Wissen rein deklarativ zu beschreiben und die algorithmische Bearbeitung dem PROLOG-internen Inferenzmechanismus zu überlassen. Der Vorteil ist, daß deklarative Aspekte einfacher zu verstehen und zu formulieren sind und man sich um die prozeduralen Details nicht zu kümmern braucht (Bratko 1987: 64). Die obengenannte semantische Lücke oder Schere zwischen der ursprünglichen Repräsentation einer Theorie (Logik, Mengenlehre, Graphentheorie, natürliche Sprache) und ihrer programmierten Form wird dadurch eindeutig reduziert. Ohne sich um den Algorithmus zu kümmern, können Modelle in einer Art "rapid prototyping" sogar unmittelbar in Regeln umgesetzt werden.
3. *Trennung Theorie von Steuerung*
   Durch die klare Trennung von Wissensbasis und Steuersystem werden theoretisch bedeutsame Annahmen und Regeln klar von anderen Programmteilen, die zur Steuerung dienen, abgehoben. Damit läßt sich die Forderung von Frijda (1967) nach deutlicher Abschottung theoretisch relevanter von theoretisch irrelevanten Teilen in wissensbasierten Modellen eindeutig besser erfüllen als in konventionellen Modellen.
4. *Begründung von Schlußfolgerungen*
   Wissensbasierte Modelle können aufgrund ihrer Architektur ihr eigenes Verhalten erklären und begründen. Für jede Schlußfolgerung läßt sich zeigen, durch welche Fakten und Regeln diese zustande kam. Mit wissensbasierten Modellen, die fähig sind, ihre eigenen Schlüsse zu rechtfertigen, lassen sich die hinter dem Modell steckenden Regeln und Annahmen jederzeit belegen und kritisieren. Diese Begründungen können benutzt werden zum Nachweis der Korrektheit der Ableitungen, zur Nachvollziehbarkeit der Lösungswege oder allgemein zur besseren Einsicht in die Funktionsweise des Modells. Der Vorwurf der Undurchsichtigkeit von Computermodellen wird damit weitgehend aufgehoben.
5. *Modularität und Flexibilität*
   Während in konventionellen Modellen das Wissen in Funktionen oder Prozeduren "verpackt" ist und Wissenselemente nur umständlich hinzugefügt bzw. entfernt werden können, impliziert in wissensbasierten Systemen die klare Schnittstelle von Wissensbasis und Steuersystem weitgehende Modularität und Flexibilität: Wissensstücke lassen sich einfach hinzunehmen (zusätzliche Regeln einführen), ändern (bestehende Regeln modifizieren) oder entfernen (Regeln streichen). Dies begünstigt zweifellos den experimentellen Charakter der Modelle.

6. *Ableitung als Deduktion und Schlußrechtfertigung als Erklärung*
   In logikbasierten Regelsystemen entspricht der Begriff der Ableitung unmittelbar dem logischen Deduktionsbegriff und der Begriff der Erklärung unmittelbar dem H-O-Schema. Eine Aufforderung an das System, ein Ereignis zu erklären, korrespondiert - unter der Bedingung, daß die Regel ein Gesetz repräsentiert - direkt dem nomologischen Erklärungsschema: die Aufforderung, einen Schluß zu erklären, löst die Ausgabe der (Gesetzes-)Regel(n) aus und der Fakten, aus denen das zu erklärende Ereignis deduziert wurde.
7. *Mechanische Deduktion*
   Die zentrale Eigenschaft und Stärke wissensbasierter Modelle ist, daß die logischen Eigenschaften einer (qualitativen) Theorie maschinell untersucht und Folgerungen automatisch deduziert werden können (Glorie/Masuch/Marx 1992: 80-82). Dieser - bereits in Kap. 2.5 erwähnte - Nutzen kommt beim wissensbasierten Ansatz wesentlich prägnanter und klarer zur Geltung als bei konventionellen numerischen Modellen. In wissensbasierten Modellen kann automatisches Ableiten die Fallen des intuitiven Schließens und die Fehler der verbalen Theoriekonstruktion eliminieren. Diese Fehler stammen primär von den Grenzen der Rechenkapazität des menschlichen Gehirns (wie dem kleinen Kurzzeitgedächtnis) und von psychologischen Verzerrungen, die den Inferenzprozeß stören. Beispielsweise wird Information selektiv verarbeitet, "wünschbare" und "interessante" Konklusionen werden leichter gezogen und Ableitungen sind von den Werten des Forschers abhängig. Im Gegensatz zum intuitiven, "gehirnbasierten" Schlußfolgern können computerbasierte Deduktionen zuverlässig, reproduzierbar und wertneutral gemacht werden. Der Computer läßt sich - anders als Menschen - weder durch psychologische Faktoren noch durch eine große Menge von Daten oder Regeln (oder beidem) verwirren. In folgenden Bereichen kann der wissensverarbeitende Computer beispielsweise besser sein als der Mensch:
   - Beim Ableiten vollständiger und korrekter Konklusionen aus einer großen Datenmenge;
   - Beim Entdecken versteckter Inkonsistenzen in verbalen Theorien;
   - Beim Prüfen, ob bestimmte Annahmen von anderen abhängen (logische Unabhängigkeit);
   - Beim Untersuchen kausal komplexer Bereiche und komplexer Interaktionen in Theorien;
   - Beim Entdecken kontraintuitiver Propositionen.

Metaphorisch gesprochen kann das wissensbasierte Modell die Rolle eines idealen Forschungsassistenten spielen: "For the expert social scientist a KBS (knowledged based system, Anm.d.Verf.) may play a role analogous to that of a good research assistent, checking the logic of the researcher, tracing out the implications of changing an assumption, answering questions about its reasoning, and identifying the source of particular bits of knowledge" (Brent 1986: 270-271).

Trotz vieler Gemeinsamkeiten zwischen Expertenwissen, Expertensystemen und Theorien gibt es allerdings auch Unterschiede. Reale Expertensysteme müssen mit viel komplexerem Wissen umgehen, als dies bei den meisten Theorien der Fall sein dürfte.

Die Wissensrepräsentationsformalismen praktisch ausgerichteter Expertensysteme gehen weit über Prädikatenlogik hinaus und beinhalten z.T. komplexe Wissensstrukturen.[42] Moderne Expertensysteme verwenden vielfach hybride - d.h. unterschiedliche - Repräsentationsformalismen innerhalb einer Problemstellung. Bei den unten behandelten Theorien hat die Datenbasis hingegen eine sehr einfache und einheitliche Struktur. Die empirische Faktenbasis besteht aus sehr vielen, aber einfachen 2-stelligen Relationen. In diesem Fall macht es keinen Sinn, das Expertensystem so zu konstruieren, daß es mit der Abarbeitung von Regeln nach Benutzerdaten fragt oder die Anfrage nach Daten in der Erklärungskomponente rechtfertigen zu lassen.

Die Stärke von Expertensystemen wird in der Literatur oft in der maschinellen Behandlung unsicheren und vagen Wissens gesehen. Haas (1990) betont z.B. gerade den Vorteil der Behandlung unsicheren Wissens für sozialwissenschaftliche Expertensysteme. Versucht man explizit vorliegende Theorien in solche Systeme zu transferieren, spielen Vagheiten, Unsicherheiten und Daumenregeln - zumindest bei den hier behandelten Modellen - keine Rolle. Für die Darstellung von Unsicherheit in sozialwissenschaftlichen Expertensystemen möchten wir auf Haas (1990) und Benfer/Brent/Furbee (1991) verweisen.

Unser Ziel ist es nicht, aus Theorien *vollständige* Expertensysteme zu machen, sondern bestimmte *Aspekte* der Expertensystem-Technologie zur Modellierung zu verwenden. Von den in Kap. 4.1.2 angesprochenen Komponenten sind für unsere Zwecke nicht alle bedeutsam. Beispielsweise werden wir auf die Installation einer Wissenserwerbs-Komponente verzichten. Wir sind auch nicht daran interessiert, möglichst benutzerfreundliche Systeme mit schönen Oberflächen zu generieren, sondern die Modelle sollen als Werkzeuge für den Wissenschaftstheoretiker und Substanzwissenschaftler zur Theorienbildung und -analyse eingesetzt werden können.

4.3.2 Wissensbasierte Modelle in den Sozialwissenschaften

Wissensbasierte Modelle wurden in den Humanwissenschaften vor allem in der kognitiven Psychologie verwendet. Bekannte Vertreter und Initiatoren dieses Ansatzes sind Allan Newell und Herbert Simon (vgl. z.B. Newell/Simon 1972). Ziel dieser KI-orientierten Modellbildung ist es, mentale Prozesse mit wissensbasierten Systemen zu "rekonstruieren" und auf diese Weise besser verstehbar zu machen. Den theoretischen Hintergrund bildet die Annahme, daß kognitive Prozesse wie Problemlösen, Planen oder Sprachverstehen als wissens- oder regelbasierte Vorgänge der Symbolverarbeitung aufgefaßt werden können. Wissensbasierte Systeme werden dabei oft direkt als psychologisch interpretierbare Modellvorstellung für die Architektur des menschlichen kognitiven Apparats angesehen (Opwis 1992: 80-86). Beispielsweise spiegeln Regeln den assoziativen Charakter weiter Bereiche menschlichen Wissens wieder und eignen sich - durch Hinzufügen bzw. Löschen von Regeln - gut zur Modellierung von Lern- und Vergessensvorgängen. Die strukturellen Bestandteile eines solchen Systems - Faktenbasis und Regeln - lassen sich als Strukturtheorien des

[42] In der KI gibt es eine Diskussion darüber, inwieweit Prädikatenlogik überhaupt zur Darstellung von Alltagsswissen geeignet ist.

Gedächtnisses interpretieren: der Faktenbasis entspricht das Kurzzeitgedächtnis, den Regeln das Langzeitgedächtnis. Bezüglich der Speicherung und des Abrufs von Informationen können eine Vielzahl gedächtnispsychologischer Vorstellungen in diesem Ansatz rekonstruiert werden. In der kognitiven Psychologie und in anderen Teilgebieten der Psychologie haben sich deshalb wissensbasierte Systeme aufgrund ihrer hervorstechenden Eigenschaften zum bevorzugten formalen Modellierungsinstrumentarium entwickelt. Ein Überblick über diesen Ansatz mit vielen Beispielen findet sich in Opwis (1992).

In den Sozialwissenschaften ist die Motivation für die Verwendung KI-basierter Modelle eine andere. Erste sozialwissenschaftlich relevante KI-Modelle entstanden bereits in den sechziger und siebziger Jahren, wobei die Rezeption dieser Modelle aber weitgehend auf die KI-Gemeinde beschränkt blieb. Sie unterscheiden sich auch insofern von dem hier betrachteten Modelltyp, als sie sich (noch) nicht explizit auf Wissen oder Regeln beziehen. Zu diesen frühen KI-Modellen zählt das S.52-54 erwähnte Programm von Gullahorn/Gullahorn (1963) zur Interaktionstheorie von Homans (1961). Ein anderes frühes Beispiel ist die IDEOLOGY MACHINE von Abelson (1973). Abelson betrachtet den ideologischen Konflikt des kalten Krieges aus der Sicht des KI-Wissenschaftlers und konstruiert ein Skript, das mentale Prozesse eines rechtsradikalen Ideologen repräsentiert. Eine detaillierte, zusammenfassende Beschreibung gibt Boden (1987).

In den achtziger Jahren erkannten und diskutierten einige Sozialwissenschaftler die Relevanz *regelbasierter* Systeme für Modellierungszwecke. Vordergründigstes Ziel wissensbasierter Modellierung ist die Möglichkeit der Formalisierung und Explikation *verbaler* theoretischer Repräsentationen, die sich einer quantitativ-mathematischen Modellierung entziehen. Beispielsweise betrachtet Brent (1986) wissensbasierte Systeme als "qualitativen Formalismus" zur Repräsentation nicht quantitativ-mathematisch formalisierbarer Theorien: "To sociologists who have found traditional mathematical models to be appropriate and useful for only a limited range of social phenomena, the greatest appeal of KBS's (knowledged based systems, Anm.d.Verf.) is their claim to be applicable to problems not having tractable solutions based on mathematical reasoning" (Brent 1986: 258). Ein Überblick über Anwendungen regelbasierter Systeme in den Sozialwissenschaften findet sich in Benfer/Brent/Furbee (1991).[43] Wir wollen einige typische Modelle betrachten.

Thorson/Sylvan (1982) benutzen ein Produktionssystem um Kennedys Entscheidungen während der Kuba-Krise zu modellieren. Kennedys Entscheidungsprozesse werden als Wenn-dann-Regeln repräsentiert, und das Programm wird zunächst mit Ereignissen getestet, die tatsächlich während der Kuba-Krise auftraten. Das Modell reagierte ähnlich wie Kennedy in der historischen Situation. Die Autoren führten anschließend Experimente durch um zu sehen, wie Kennedy bei Ereignismengen reagiert hätte, die historisch nicht auftraten.

---

[43] Benfer/Brent/Furbee (1991) geben ebenfalls einen Überblick über Verwendungsmöglichkeiten sozialwissenschaftlicher Expertensysteme *außerhalb* von Modell- und Theorienbildung im engeren Sinn. Hierzu gehören z.B. statistische Expertensysteme zur Datenanalyse oder Systeme zur Unterstützung der Auswahl bestimmter Forschungsmethoden.

Sylvan/Glassner (1985) benutzen ein Logikprogramm, um theoretische Aussagen von Simmel (1955) zu prüfen und übersetzen 107 Passagen aus Simmels "Conflict and the Web of Group Affiliations" in 223 Regeln. Es wird geprüft, ob aus der Theorie Kontradiktionen ableitbar sind und ob es möglich ist, die zentrale Behauptung von Simmel, nämlich die Konflikt-Kohäsions-Hypothese, zu deduzieren. Es zeigt sich, daß aus den formalisierten Regeln zwar keine Kontradiktionen abgeleitet werden können, aber ebensowenig kann die zentrale Hypothese bewiesen werden. Mit dem Programm kann also belegt werden, daß die - übersetzten - verbalen Annahmen von Simmel nicht hinreichend sind, um seine zentrale Behauptung deduzieren zu können.

Banerjee (1986) versucht mit Hilfe eines PROLOG-Programms die Entstehung von sozialen Strukturen zu erklären. Sozialstruktur ist für Banerjee einfach ein Muster von Wiederholungen sozialer Handlungen. Zur Erklärung sozialer Strukturen benutzt Banerjee Schema-Konzepte aus der KI, in denen die Muster sozialer Handlungen gespeichert sind. Die Annahme ist, daß Akteure in sozialen Situationen passende Schemata suchen, aktivieren und dann ausführen. Diese Konzepte werden in einem PROLOG-Programm operationalisiert. Benfer/Brent/Furbee (1991: 26) weisen darauf hin, daß in diesem Programm eine Theorie entstanden ist, die vorher nicht explizit vorhanden war: "Banerjee's work is an example of the use of artificial intelligence programming strategies where the AI program becomes the theory itself".

Die Forschungsgruppe um Michael Masuch (Center for Computer Science in Organization and Management, Amsterdam) übersetzt verschiedene organisationssoziologische Theorien in Prädikatenlogik und Clausenform. Unter Nutzung eines Theorembeweisers können alle logisch möglichen Schlußfolgerungen aus den Annahmen abgeleitet werden. Glory/Masuch/Marx (1990) präzisieren auf diese Weise die Struktur von Mintzbergs Kontingenztheorie und entdecken während des Formalisierungsprozesses an unterschiedlichen Stellen terminologische Probleme. Ein anderes Beispiel aus dieser Gruppe (Péli/Bruggeman/Masuch/Nualláin 1992) ist die prädikatenlogische Übersetzung der Organisationsökologie von Michael Hannan und John Freeman. Die Anwendung des Theorembeweisers zeigt, daß der modellierte Teil - mit einer Ausnahme - konsistent ist. Weiter werden eine Reihe neuer Theoreme entdeckt, während einige postulierte Konklusionen aus den Annahmen nicht hergeleitet werden können.

Die Beispiele betonen verschiedene Seiten und Ziele wissensbasierter Modellierung (vgl. Benfer/Brent/Furbee 1991): Thorson/Sylvan (1982) *simulieren* tatsächliche und fiktive historische *Ereignisse*, Sylvan/Glassner (1985) und die Masuch-Gruppe prüfen die *logische Konsistenz und Vollständigkeit existierender Theorien* und in Banerjee (1986) *entsteht mit dem Programm eine neue Theorie.* Ein wichtiger inhaltlicher Anreiz für die regelbasierte Darstellung ist insbesondere auch die *qualitative Modellierung von Klassikern der Soziologie*, wie sie in Sylvan/Glassner (1985) und dem folgenden Beispiel erfolgt. Diese Modelle, sowie die der Masuch-Gruppe, kommen unserem folgenden Ansatz am nächsten.

Brent (1986) will am Beispiel einer klassischen, aber hochgradig "dunklen Theorie" - Goffmans Interaktions-Modell des "dramaturgical framework" - demonstrieren, daß ein weiter Bereich umgangssprachlich-theoretischer Propositionen in PROLOG-

Clausen übersetzt werden kann. Er transferiert einen Teil des Goffman-Textes - der "bislang allen Formalisierungsversuchen widerstand" - nach PROLOG um zu demonstrieren, daß der nach PROLOG übersetzbare Bereich sozialwissenschaftlichen Wissens größer ist als der für traditionelle mathematische Modelle und weniger Strukturierung verlangt. Der folgende Auszug ist ein Beispiel, wie eine verbale Definition in eine PROLOG-Regel übersetzt wird (Brent 1986: 264).

Tab. 4.3: Eine umgangssprachliche Proposition von Goffman und die entsprechende Repräsentation als PROLOG-Regel.

Umgangssprache	*PROLOG*

performance refer(s) to all the activity of an individual which occurs during a period marked by his continuous presence before a particular set of oberservers and which has some influence on the observers.	`is_a(X,performance) :-`
	`  is_a(X,activity),`
	`  occurs_during(X,Period),`
	`  marked_by(Period),`
	`  contin_pres_of(Obsvs),`
	`  influences(X,Obsvs).`

Brent überträgt die Kernsätze der umgangssprachlichen Theorie direkt in PROLOG. Dieses Vorgehen ist typisch für die meisten Modellbauer. Ein unmittelbares, lineares Satz-für-Satz-Übersetzen von der Umgangssprache in eine formale Sprache erscheint aber problematisch. Verbale Theorien - und hier ist die Goffman-Theorie ein Beispiel par excellence - sind oft in einer blumig-vagen und hochambigen Prosa gehalten, die bei der direkten Übertragung in das Programm ihre Mängel in die Formalisierung mitschleppen: "Moving from sentence to sentence, trying to find some translation for each phrase, one would end up with an uncontrollable number of concepts in the formal language. Many of these concepts would overlap, reproducing the ambiguities of the natural language" (Glorie/Masuch/Marx 1990: 85-86).

In dieser Arbeit wird deshalb die Auffassung vertreten, daß man vor der Implementierung den logischen Aufbau der Theorie explizit machen und klären sollte. Konkret empfiehlt sich, sich zunächst einmal Gedanken zu machen über die Grundbegriffe, die verwendeten Relationen und deren Stelligkeit, sowie die postulierten Zusammenhänge, um dann zu unterscheiden zwischen Definitionen, Gesetzen, Daten und Theoremen. In PROLOG ist alles in Fakten und Regeln zu formalisieren, so daß z.B. PROLOG-intern zwischen Definitionen und Gesetzen kein Unterschied besteht. Die Aussage von Goffman in Tab. 4.3 ist offensichtlich eine Definition, was aber weder in der umgangssprachlichen Darstellung noch in PROLOG expliziert wird. Eine logische Rekonstruktion der Theorie würde dies explizit offen legen. Die strukturalistische Wissenschaftstheorie leistet nun genau die gewünschte logische Explikation empirischer Theorien, auf deren Basis die Computermodellierung erfolgen kann.[44]

[44] Im übrigen wird auch in kommerziellen und praxisnahen Expertensystemen das Wissen des Experten normalerweise nicht unmittelbar in die Maschine transferiert. Der "Wissenstransfer-Experte" oder Knowledge-Engineer strukturiert und "rekonstruiert" vielmehr erst das Expertenwissen, um es dann systematisch aufbereitet zu implementieren.

5. Strukturalistische Theorienkonzeption und Wissensverarbeitung

Aus KI-Sicht kann das strukturalistische Theorienkonzept als Repräsentationsschema für empirisches, wissenschaftliches Wissen betrachtet werden. Dieses empirisch-wissenschaftliche Wissen bildet in Absetzung zum bloßen Glauben oder anderen Wissensformen *das* herausgehobene und paradigmatische Beispiel für Wissen. Es findet sich in Lehrbüchern und Zeitschriften der einzelnen Wissenschaftsdisziplinen.

Nach Auffassung der Strukturalisten ist die linguistische Interpretation wissenschaftlichen Wissens als "Menge von allgemeinen Propositionen" viel zu schwach und unspezifisch, um wissenschaftliche Theorien zu charakterisieren. Das allgemeine Reden über Alltagserfahrungen erzeugt sicherlich keine empirische Theorie (Sneed 1979: 2). Im Verständnis der strukturalistischen Wissenschaftstheorie liegt das wesentliche Merkmal empirisch-wissenschaftlichen Wissens nicht in der propositionalen Natur, sondern in der hochkomplexen Struktur oder "Architektonik" dieses Wissens. Das strukturalistische Programm expliziert diese Wissensarchitektonik und bietet es als metatheoretischen, standardisierten Repräsentationsformalismus an (Balzer/Moulines/Sneed 1987). Bevor die Grundzüge dieses Konzepts dargestellt werden, sollen kurz die Hintergründe seines Entstehens angedeutet werden.

5.1 Zur Krise des Standardtheorienkonzepts

Seit den sechziger Jahren befindet sich das Standardtheorienkonzept, insbesondere die Methodologie des Kritischen Rationalismus, in einer tiefen Krise. Differenzierte Untersuchungen zur Geschichte der Naturwissenschaften haben ergeben, daß die Entwicklung der naturwissenschaftlichen Disziplinen nicht im geringsten der Popperschen Falsifikationsmethodologie folgt. Wie die historischen Analysen zeigten, ist für weite Perioden naturwissenschaftlicher Forschung nicht der Versuch charakteristisch, Theorien zu widerlegen, sondern diese zu bestätigen und zu verteidigen - auch gegen jegliche empirische Evidenz.

Bahnbrechend für die Kritik des Kritischen Rationalismus war die Arbeit des Wissenschaftshistorikers Thomas S. Kuhn (1991, zuerst 1962): "The Structure of Scientific Revolutions". Kuhn wandte sich gegen die bis dato allgemein akzeptierte Vorstellung, nach der (Natur-)Wissenschaft ein rationaler, kumulativer Prozeß sei, bei dem Theorien an der Erfahrung scheitern und durch neue, erfolgreichere und bessere Theorien ersetzt werden. Kein einziger, von Kuhn aufgedeckter Ablauf hatte auch nur die geringste Ähnlichkeit mit der Popperschen Falsifikationsschablone. Wenn alte Theorien von neuen Theorien verdrängt wurden, dann *ohne* Dazwischenschaltung von Erfahrung, oft durch "irrationale" Mittel der Überredung, Propaganda oder Tod einer Forschergeneration. Bei dieser "außerordentlichen" Form der Wissenschaft handelt es sich um traditionszerstörende Ersetzungen und "wissenschaftliche Revolutionen" im Gegensatz zur "normalen Wissenschaft", in der Wissenschaftler ihre Tätigkeit im

Rahmen einer bestimmten wissenschaftlichen Tradition - eines "Paradigmas" - verrichten (Stegmüller 1979: 737-738). In der normalen Wissenschaft halten Forscher an einer Theorie fest, auch wenn Anomalien auftreten und die Theorie nach dem Falsifikationskriterium eigentlich aufzugeben wäre. Mit seinen provozierenden, aber zum Großteil zutreffenden Analysen der wissenschaftlichen Entwicklungsprozesse leitete Kuhn eine historisch-pragmatische Wende in der Wissenschaftstheorie ein.[45]

Die idealisierend-normative Ausrichtung der analytischen Wissenschaftstheorie und die damit verbundene, immer größer werdende Kluft zur realen Wissenschaft zeigte sich auch in anderen Bereichen. Ein Beispiel sind die miteinander rivalisierenden induktiven Bestätigungs- und deduktiven Bewährungstheorien. Ihren Kritikern zufolge werden in diesen Metatheorien nicht faktische Vorkommnisse in den Einzelwissenschaften unter einem idealisierenden Aspekt betrachtet, und den Wissenschaftlern werden Argumentationsweisen unterstellt, die es niemals auch nur in Ansätzen gegeben hat (Stegmüller 1973: 4).

Die klare Diskrepanz zwischen dem faktischen Verlauf wissenschaftlicher Forschung und den postulierten wissenschaftstheoretischen Rationalitätsnormen bildete jedoch nicht den unmittelbaren Auslöser für die Entwicklung der strukturalistischen Wissenschaftstheorie. Als wesentliche Triebfeder für die Ausarbeitung der neuen Wissenschaftstheorie wird in der Literatur die Kritik am Zweistufenkonzept der Wissenschaftssprache betrachtet. Diese Kritik äußerte sich in verschiedenen Formen. Zum Teil richtete sie sich gegen den Begriff der Beobachtungssprache selbst, zum Teil wurde die Eindeutigkeit der Dichotomie "beobachtbar-theoretisch" bezweifelt. Es machte sich mehr und mehr die Überzeugung von der "Theorienbeladenheit aller Beobachtungen" und der Nichtexistenz einer neutralen, theorieunabhängigen Beobachtungssprache breit (Stegmüller 1973: 27-34). Insbesondere die genaue Rolle, die theoretische Terme in einer Theorie spielen, war ungeklärt und letztendlich Auslöser für den Strukturalismus. Alle bisherigen Versuche, theoretische von nichttheoretischen Begriffen abzugrenzen, waren von Negativität gekennzeichnet in dem Sinn, daß ein Begriff theoretisch ist, weil er nicht zur Beobachtungssprache gehört. Tatsächlich scheint jedoch in vielen Fällen etwas wesentlich Stärkeres intendiert zu sein, nämlich, daß ein Term bezüglich einer Theorie deshalb theoretisch ist, weil er im Rahmen der Theorie eine ganz bestimmte Stellung einnimmt, die ihn von jenen Termen, die diese Stellung nicht einnehmen, scharf unterscheidet. Hilary Putnam hat diese Herausforderung prägnant wie folgt formuliert: was genau hat ein theoretischer Term, der mit Recht so genannt wird, mit der wissenschaftlichen Theorie, in der er vorkommt, zu tun? Was also sind die auszeichnenden Merkmale theoretischer Terme? (Stegmüller 1973: 31).

---

[45] Mit der historisch-pragmatischen Wende wurde von dem Dogma Abschied genommen, Wissenschaftstheorie könne allein mit den syntaktischen und semantischen Werkzeugen der modernen Logik betrieben werden. Stattdessen traten zunehmend wissenschaftshistorische, -psychologische und -soziologische Aspekte in den Vordergrund. Kuhn war für viele Wissenschaftsphilosophen und Substanzwissenschaftler der Anlaß, sich endgültig von den Programmen der Logischen Empiristen und Kritischen Rationalisten zu lösen. Eine ausführliche Rezeption des Werks von Kuhn findet sich z.B. in Stegmüller (1979).

5.2 Grundzüge der strukturalistischen Theorienkonzeption

Das strukturalistische Theorienkonzept kann zur Diskussion um die Krise des Standardtheorienkonzepts wesentliche Beiträge und Lösungen anbieten und erlaubt es, empirische Forschung erheblich differenzierter zu betrachten als der Kritische Rationalismus oder Logische Empirismus. Es basiert auf Ideen von Patrick Suppes und wurde von Joseph Sneed (1971) in seinem Buch "The Logical Structure of Mathematical Physics" ausgearbeitet. Stegmüller widmete in den siebziger und achtziger Jahren einen Großteil seines Schaffens der Vereinfachung, Präzisierung und Popularisierung des Programms von Sneed. Beide entwickelten den Ansatz zusammen mit Balzer und anderen weiter. Die strukturalistische Deutung von Theorien wurde ursprünglich in Absetzung zum nicht mehr für adäquat gehaltenen Aussagenkonzept als Nicht-Ausssagenkonzept ("non-statement view") bezeichnet und später umbenannt. Der Grund für die Bezeichnung "Strukturalismus" liegt darin, daß Theorien nicht mehr als System von Sätzen gedeutet werden, sondern als Gebilde, deren wichtigste Bestandteile mathematische Strukturen sind (Stegmüller 1979: 752).

Ein charakteristischer Grundzug der strukturalistischen Theoriensicht ist ihre liberalere und pragmatischere Auffassung von Wissenschaftstheorie. Dies bezieht sich insbesondere auf drei Aspekte. Erstens beschränkt sich die strukturalistische Wissenschaftstheorie nicht auf eine rein formale Analyse von Theorien, sondern erlaubt den Einbezug wissenschaftshistorischer, -psychologischer und -soziologischer Aspekte. Zweitens ist das strukturalistische Programm - anders als der Logische Empirismus und Kritische Rationalismus - nicht normativ orientiert. Das Interesse der strukturalistischen Schule gilt vielmehr der *rationalen Rekonstruktion* oder *Explikation* bestehender Theorien. Damit ist gemeint, daß die *vorgegebenen* Konzepte einer Theorie (die *präsystematisch* vorgegebene Theorie) durch ähnliche, aber klarere, exaktere, konsistentere oder fruchtbarere ersetzt und dadurch Inkonsistenzen und Ungenauigkeiten beseitigt werden (Westermann 1987: 6). Die im Verhältnis zu seinen Vorläufern "bescheidene" Funktion des Strukturalismus ist also eher systematisierend und beschreibend, allenfalls kommt dem Strukturalismus noch die Rolle eines Korrektivs zu, keinesfalls aber die eines Normgebers (Westermann 1987: 7). Ein drittes Merkmal für die pragmatische und liberale Auffassung von Wissenschaftstheorie ist die Abkehr vom formalsprachlichen Vorgehen bei der Rekonstruktion von Theorien. Die von Carnap vertretene formalsprachliche Methode bedeutete eine vollständige Formalisierung einer Theorie in einer Kunstsprache, z.B. einem formalen mengentheoretischen System. Das strukturalistische Theorienkonzept hingegen legt bei der Rekonstruktion kein formales mathematisches System zugrunde, sondern nur *informelle* Logik und Mengenlehre. In der von Stegmüller benutzten - scheinbar widersprüchlichen - Bezeichnung des strukturalistischen Vorgehens als "informelle Formalisierung" soll ausgedrückt werden, daß Theorien zwar formal dargestellt werden, aber dieser Darstellung keine formale Kunstsprache zugrunde liegt. Logische Konstanten wie $\wedge$ ("und"), $\vee$ ("oder"), $\neg$ ("nicht"), $\rightarrow$ ("wenn...dann"), $\forall$ (Allquantor) und $\exists$ (Existenzquantor) sind nicht als Zeichen einer formalen Sprache aufzufassen, sondern als Abkürzungen für die entsprechenden umgangssprachlichen

Ausdrücke, allerdings mit den bekannten Normierungen im Fall des wenn-dann (Stegmüller 1986: 21).

5.2.1 Informelle Axiomatisierung

Die Ausgangsidee der strukturalistischen Theorieninterpretation läßt sich auf Patrick Suppes zurückführen. Unter Verweis auf das sog. Bourbaki-Programm in der Mathematik schlug Suppes vor, an die Stelle formalsprachlicher Methoden informelle mengentheoretische Axiomatisierungsverfahren zu setzen.[46] Die Axiomatisierung einer gegebenen Theorie geschieht dabei durch *Definition eines mengentheoretischen Prädikats* der Form "... ist ein P". Die mengensprachlich formulierten Axiome bilden bei diesem Vorgehen den Definitionsbestandteil des eingeführten Prädikats. In der Mathematik wird z.B. die Gruppentheorie oder die Wahrscheinlichkeitstheorie dadurch axiomatisiert, daß ein mengensprachliches Prädikat "...ist eine Gruppe" oder "... ist ein Wahrscheinlichkeitsraum" eingeführt wird. Die sogenannten Axiome sind nichts anderes als bestimmte Bestandteile im Definiens des fraglichen Prädikats.

In einer kühnen, aber sehr naheliegenden Verallgemeinerung, übertrug Suppes diesen Gedanken auf empirische Theorien der mathematischen Physik. "Die klassische Partikelmechanik zu axiomatisieren" heißt dann, das mengentheoretische Prädikat "... ist eine klassische Partikelmechanik" zu definieren. Dieses Prädikat wird mit Hilfe der grundlegenden physikalischen Gesetze oder Axiome von Newton definiert, nämlich dem Trägheitsgesetz, dem dynamischen Grundgesetz (Kraft = Masse * Beschleunigung) und dem Actio-Reactio-Prinzip (vgl. Balzer/Moulines/Sneed 1987: 103-108). In analoger Weise bedeutet "die Quantenmechanik zu axiomatisieren", das mengentheoretische Prädikat "... ist eine Quantenmechanik" zu definieren und im Definiens die grundlegenden Gesetze der Quantenmechanik festzulegen. (Stegmüller 1980: 5).

Um den Leser nicht mit physikalischen Theoriedetails zu verwirren, wählen wir zur Illustration der Suppes-Methode den Strukturbegriff (die informelle Axiomatisierung läßt sich nicht nur mit Theorien durchführen, sondern mit beliebigen Begriffen). Der Strukturbegriff dient ja nicht nur als neuere Bezeichung des hier behandelten Nicht-Aussagenkonzepts von Theorien, sondern wird insbesondere in den Sozialwissenschaften in schillernder Weise verwendet. Eine wesentliche Rolle spielt der Strukturbegriff auch bei den im zweiten Teil behandelten Theorien. Die folgende Definition legt den Term "Struktur" durch Einführung des mengentheoretischen Prädikats "... ist eine Struktur" fest (nach Balzer 1982: 273).

---

[46] Die "Bourbaki"-Gruppe war ein Zusammenschluß einer Reihe von Mathematikern nach dem zweiten Weltkrieg. Sie verfolgte das Ziel eines präzisen Aufbaus der modernen Mathematik. Das Bourbaki-Programm forderte - ähnlich wie Hilbert (vgl. Kap. 2.2) - ein klares axiomatisches Gerüst für jedes mathematische Teilgebiet, in dem scharf unterschieden wird zwischen Grundbegriffen, Definitionen, Axiomen und Theoremen. Die Mitglieder der Gruppe hatten wohlweislich darauf verzichtet, die Formalisierung der Mathematik in einer Kunstsprache vorzunehmen. Der Grund ist, daß so etwas außer für metamathematisch Interessierte nicht nötig sei und daß es außerdem die Realisierung des Projekts um mindestens hundert Jahre verzögert hätte. Das Programm ist mittlerweile zu einem großen Teil verwirklicht (Stegmüller 1979: 469-470).

Definition
x ist eine Struktur gdw es $M_1,...,M_n$ und $R_1,...,R_k$ gibt, so daß gilt:
(1) $x = \langle M_1,...,M_n;R_1,...,R_k\rangle$
(2) $M_1,...,M_n$ sind nicht-leere Mengen
(3) Für alle $i \in \{1,...,k\}$ gilt: R_i ist eine Relation über $M_1,...,M_n$

In dieser Definition ist "Struktur" der zu definierende Begriff, der durch die mengensprachlichen Axiome (1)-(3) bestimmt wird. Die Axiome (1)-(3) sind dabei Bestandteile im Definiens des Definiendum "Struktur". Sie besagen (in dieser Reihenfolge), daß eine Struktur ein (n+k)-Tupel ist mit n nicht-leeren Mengen und k Relationen über diesen Mengen. Eine Struktur ist also informell gesprochen einfach eine Anzahl von Mengen mit bestimmten Relationen über den Mengen.

Nachdem der Begriff der Struktur explizit definiert wurde, *wird man eine Entität dann und nur dann als Struktur bezeichnen, wenn die Axiome (1) bis (3) erfüllt sind.* Das mengensprachliche Prädikat "... ist eine Struktur" läßt zunächst offen, aus welcher Art von Objekten die Mengen M_i bestehen und wie die einzelnen Relationen R_i definiert sind. Bei den im zweiten Teil behandelten balancetheoretischen Strukturen wird M (mit n, k = 1) beispielsweise unter anderem als Menge von Personen und $R \subseteq M \times M$ als 2-stellige Freundschaftsrelation interpretiert. Ist $x, y \in M$ und gilt xRy, so würde das in dieser Interpretation bedeuten, daß x Freund von y ist. Analog kann R als Machtrelation, Kommunikationsrelation usw. interpretiert werden. Ein Beispiel für eine 3-stellige Relation $R \subseteq M_1 \times M_1 \times M_2$ mit M_1 als Menge von Personen und M_2 als Menge "menschlicher Fähigkeiten" wäre: x erkennt y bezüglich z an, wobei $x, y \in M_1$ und $z \in M_2$. Wie die Interpretation auch sei, die mengensprachliche Strukturdefinition ist ein einfaches Beispiel für die Axiomatisierung eines sozialwissenschaftlich bedeutsamen Begriffs nach der Methode von Suppes.

5.2.2 Der mathematische Theoriekern

Suppes Vorschlag, die informelle mengentheoretische Axiomatisierung auch auf empirische Theorien anzuwenden, wurde von Sneed (1971) aufgegriffen. Sneed, der eigentliche Begründer des strukturalistischen Programms, ergänzte und präzisierte den Ansatz von Suppes mittels informeller Semantik und Modelltheorie.

Eine vorexplikativ gegebene Theorie T informell mengentheoretisch zu axiomatisieren bedeutet - wie eben dargelegt - das die Theorie ausdrückende, mengentheoretische Prädikat "... ist ein P" einzuführen. Alle Entitäten, welche das Prädikat P erfüllen, heißen *Modelle der Theorie T.* Modelle sind nichts anderes als der Begriffsumfang (die Extension) oder die "Wahrheitsfälle" des entsprechenden Prädikats P und werden mit "M" bezeichnet. Ob man vom Prädikat "P" oder von der korrespondierenden Menge "M" der Modelle dieser Theorie spricht, läuft auf dasselbe hinaus, nur daß man sich im ersten Fall auf eine linguistische Einheit bezieht, im zweiten Fall dagegen auf deren Umfang (Stegmüller 1980: 5). In der Definition des Strukturbegriffs nennt man z.B. jede Entität, welche die Axiome (1) bis (3) erfüllt, ein *Modell* des definierten Prädikats, und die Menge M der Modelle ist einfach die Extension des

Prädikats "... ist eine Struktur", also die Menge aller Entitäten, auf die das Prädikat zutrifft. Die linguistische Sprechweise "... ist ein P" ist also der modelltheoretischen Redeweise "... ist ein Modell von T" äquivalent.[47]

Bei empirischen Theorien bilden die Axiome, welche theoretische Zusammenhänge zwischen den Grundbegriffen festlegen, den wesentlichen Bestandteil der Modelldefinition. Axiome, die theoretische Zusammenhänge zwischen den Grundbegriffen fixieren, nennt man *"eigentlich inhaltliche" Axiome*. Sie drücken das *Fundamentalgesetz* einer Theorie aus (im Fall mehrerer Gesetze ist das Fundamentalgesetz die konjunktive Verknüpfung der einzelnen Axiome). Alle und nur diejenigen Entitäten, welche das Fundamentalgesetz erfüllen, sind Modelle der Theorie.

Von den Modellen einer Theorie werden in der strukturalistischen Theorienkonzeption als zweite mengentheoretische Struktur die *potentiellen Modelle* unterschieden. Die Menge der potentiellen Modelle bezeichnet man als M_p. Mögliche oder potentielle Modelle entstehen aus Modellen, indem die eigentlich inhaltlichen Axiome weggelassen werden. Von potentiellen Modellen wird also nicht verlangt, daß sie die inhaltlichen Axiome wirklich erfüllen; sie definieren lediglich das Begriffsgerüst einer Theorie. Die Hinzunahme der eigentlichen Axiome ergänzt das potentielle Modell M_p zu einem Modell M.

Gegenstand des Interesses des Theoretikers ist grundsätzlich nur das, was sich mit den Begriffen des potentiellen Modells *beschreiben* läßt. Potentielle Modelle charakterisieren somit diejenigen Entitäten, von denen es überhaupt *sinnvoll* ist zu fragen, ob sie das Prädikat "... ist ein Modell von T" erfüllen oder nicht. Die potentiellen Modelle bilden damit die größere Menge von Entitäten: zwischen Modell und potentiellem Modell besteht die Teilmengenrelation, so daß jedes Modell ein potentielles Modell ist:
$M \subseteq M_p$.

Die eben dargelegte abstrakte Charakterisierung soll an einem natur- und sozialwissenschaftlichen Beispiel verdeutlicht werden. In der klassischen Partikelmechanik bestehen die Modelle aus Systemen von Teilchen, die mit Kräften und Massen ausgestattet sind und die außerdem die Newtonschen Gesetze erfüllen. In den meisten strukturalistischen Darstellungen wird nur die Gültigkeit des zweiten Newtonschen Gesetzes gefordert (Kraft = Masse * Beschleunigung), die anderen Axiome sind bereits Spezialgesetze (vgl. Kap. 12). In diesem Verständnis besteht die informelle Axiomatisierung der Theorie in der Definition des mengentheoretischen Prädikats "... ist ein Modell der klassischen Partikelmechanik", wobei im Definiens das zweite

[47] Es sei an dieser Stelle nochmals ausdrücklich darauf hingewiesen, daß der Modellbegriff im strukturalistischen Theorienkonzept im präzisierten Sinn der modernen Logik und mathematischen Modelltheorie zu verstehen ist, wie er bereits in Kap. 1.2 eingeführt wurde. Dieser Modellbegriff unterscheidet sich in der Regel vom Modellbegriff in informellen Kontexten empirischer Wissenschaft. Während empirische Wissenschaftler dahin tendieren, "Modell" im Sinn eines "Bildes" zu verwenden, benutzen Logiker und Mathematiker umgekehrt "Modell" im Sinn der Dinge, die von einem Bild (= einer Theorie) dargestellt werden. Statt also zu sagen, daß bestimmte Gleichungen ein Modell subatomarer oder ökonomischer Phänomene sind, sprechen Formalwissenschaftler davon, daß die subatomaren oder ökonomischen Phänomene Modelle der Gleichungen sind, welche die Theorie repräsentieren. Der mathematische Modellbegriff hat den Vorteil, daß er klar definiert und gut etabliert ist (Balzer/Moulines/Sneed 1987: 2).

Newtonsche Axiom enthalten ist. Alle Entitäten, die als Systeme von Teilchen mit Kräften und Massen beschreibbar sind *und* zusätzlich das inhaltliche Axiom von Newton erfüllen, sind damit Modelle der klassischen Partikelmechanik. Beispiele für solche Modelle wären das Planetensystem, das Teilsystem Erde-Mond oder frei fallende Körper an der Erdoberfläche. Die potentiellen Modelle bilden die viel größere Gesamtheit der mit Massen und Kräften ausgestatteten Systeme von Teilchen, die *nicht* notwendig das zweite Gesetz von Newton erfüllen (Stegmüller 1979: 479). Ein Mückenschwarm könnte beispielsweise als potentielles Modell der klassischen Partikelmechanik betrachtet werden, da er ein solches Teilchensystem ist. Allerdings ist dieser kein Modell der Theorie, da das zweite Axiom hier nicht erfüllt ist.

Das eigentlich inhaltliche Axiom oder Fundamentalgesetz der Balancetheorie von Heider besagt, daß bestimmte kognitive Strukturen zur Balance tendieren (für die genaue Bedeutung dieser Aussage vgl. Kap. 6.1, S.118ff.). Die informelle Axiomatisierung dieser Theorie besteht in der Definition des mengentheoretischen Prädikats "... ist ein Modell der Heider-Theorie", wobei im Definiens der Definition das inhaltliche Axiom enthalten ist. Alle menschlichen Individuen, die diese Definition erfüllen - also alle Personen, deren kognitive Strukturen zu Balance tendieren - sind Modelle der Balancetheorie. Potentielle Modelle wären dann Personen mit kognitiven Strukturen, die in der Begrifflichkeit der Theorie beschreibbar sind, in denen aber das inhaltliche Axiom nicht unbedingt gelten muß.

Neben den Modellen und potentiellen Modellen gibt es als dritte Struktur in der strukturalistischen Wissenschaftstheorie die *partiellen potentiellen Modelle*, kurz: *Partialmodelle*. Diese entstehen aus den potentiellen Modellen, indem alle T-theoretischen Größen eliminiert werden. Mit den T-theoretischen Termen hat es folgende Bewandtnis. Die Behandlung theoretischer Begriffe war im Logischen Empirismus ein beträchtliches Problem, bildete den zentralen Kritikpunkt und letztendlich den Auslöser für die Ausarbeitung des Strukturalismus. Der Strukturalismus behandelt theoretische Begriffe ganz anders als das Aussagenkonzept: ein Term ist nicht an sich entweder theoretisch oder nicht-theoretisch - wie im Programm von Carnap - sondern er ist *theoretisch relativ zu einer Theorie T*, was mit T-theoretisch ausgedrückt wird. Ein Term ist nach Sneed genau dann theoretisch relativ auf eine Theorie T, wenn ihre Messung stets die Gültigkeit von T voraussetzt (Stegmüller 1986: 33). Theoretische Terme erhalten ihre Bedeutung typischerweise erst durch die Theorie, welche diese Terme benutzt. Beispielsweise hat der Begriff des "Unbewußten" ohne Kenntnis der Freudschen Neurosentheorie eine unklare oder gar keine Bedeutung, so daß das "Unbewußte" ein bezüglich der Freudschen Neurosenlehre theoretischer Begriff ist (Balzer 1982: 34ff.). Die Abgrenzung T-theoretischer und nicht-T-theoretischer Größen setzt eine genaue Kenntnis der Theorie voraus. Partielle potentielle Modelle sind nun nichts anders als die um T-theoretische Größen reduzierten potentiellen Modelle. Die Menge dieser partiellen potentiellen Modelle bezeichnet man mit M_{pp}.

Theoretische Terme sind der erste Grund für die Immunität von Theorien gegen Widerlegung: kommen in Gesetzen T-theoretische Terme vor, so sind diese Gesetze

unwiderlegbar.[48] Für den Zweck einer solchen Widerlegung müßte man die Terme nämlich unabhängig von der Theorie bestimmen können, was wegen der T-Theoretizität gerade nicht möglich ist (Stegmüller 1986: 200-201). In der klassischen Partikelmechanik ist beispielsweise die Unwiderlegbarkeit des zweiten Axioms immer wieder diskutiert worden, was damit begründet wurde, daß das Gesetz eine analytische Wahrheit oder Definition wäre. Strukturalisten erklären die Unwiderlegbarkeit des zweiten Axioms mit der T-Theoretizität von Masse und Kraft (Stegmüller 1980: 44-45).

Für die Sozialwissenschaften ist die Definition theoretischer Terme zu unflexibel und problematisch, da in sozialwissenschaftlichen Theorien alle Begriffe T-theoretisch sein können. Balzer (1985) ändert die Definition des partiellen potentiellen Modells ab und legt sie so allgemein fest, daß die Diskussion um theoretische Terme vermieden werden kann. Vereinfacht gesagt, besagt diese neue Theoretizitätsdefinition, "daß ein Term t in einer Theorie T theoretisch ist genau dann, wenn er in einer genau festgelegten Weise in T meßbar oder bestimmbar ist" (Balzer 1985: 139). Die Bestimmung erfolgt dabei durch eine "invariante Meßmethode", auf die wir an dieser Stelle jedoch nicht näher eingehen können.[49]

Die drei eingeführten Modellklassen - M, M_p und M_{pp} - beschreiben nun mit den in den Axiomen festgelegten Eigenschaften die mathematische Struktur einer Theorie und bilden den sog. *Theoriekern*.

5.2.3 Intendierte Anwendungen

Der Theoriekern ist nur eine formale, mathematische Struktur, die nichts über die Welt aussagt, insbesondere auch nicht, *was* überhaupt von der Welt erfaßt werden soll. Anders als *mathematische* Theorien wollen *empirische* Theorien aber Informationen über bestimmte Realitätsausschnitte liefern. Dies bedeutet, daß der Theoriekern in Beziehung gesetzt werden muß zu dem Weltausschnitt, den die Theorie behandeln soll. Diese für die Theorie vorgesehenen Realitätsausschnitte bezeichnet man als *Menge der intendierten Anwendungen* und benutzt dafür das Symbol I.

Die Menge der intendierten Anwendungen läßt sich nicht rein formal definieren und charakterisieren wie der Theoriekern. Vielmehr enthält I einfach eine Reihe realer Beispiele, die nach der sog. "paradigmatischen Methode" bestimmt werden (Stegmüller 1986: 27-28). Die "paradigmatische Methode" besagt folgendes. Es wird zunächst eine Menge I_0 ausgezeichnet, welche eine vom Begründer oder von den Begründern der Theorie genannte endliche Menge von Beispielen explizit angibt und für die eine erfolgreiche Anwendung gelungen ist. I setzt sich dann zusammen aus der Menge I_0 und einer sukzessive erweiterten Menge I* von realen Systemen, die denen

---

[48] Stegmüller (1986: 80-81) verweist auf insgesamt fünf Ursachen für die Immunität von Theorien gegen Widerlegung.

[49] Während die Klärung der Rolle theoretischer Terme ein Kernproblem der strukturalistischen Wissenschaftstheorie war, stellte sich mit deren Ausarbeitung quasi als Nebenprodukt heraus, daß auch Kuhns "Irrationalitätsthese" rational im Rahmen der neuen Metatheorie rekonstruiert werden kann. Auf diesen Aspekt kann in dieser Arbeit nicht eingegangen werden. Es sei verwiesen auf Stegmüller (1986).

von I_0 "*hinreichend ähnlich*" sind: $I := I_0 \cup I^*$. Die Ähnlichkeitsrelation läßt sich hierbei nicht präzise angeben. Vielmehr ist diese Relation als Familienähnlichkeit im Sinn von Wittgenstein aufzufassen (Wittgenstein diskutiert Familienähnlichkeiten in den "Philosophischen Untersuchungen" am Beispiel des Begriffs "Spiel"). Steht eine mögliche Erweiterung der Menge I zur Diskussion, läßt man vielfach den Theoriekern selbst bestimmen, ob die Erweiterung erfolgen soll oder nicht. Die Extension von I wird einfach durch das im Theoriekern vorkommende inhaltliche Axiom festgelegt. Da der Theoriekern in diesem Fall seine Anwendungen selbst bestimmt, spricht man von der "Regel der Autodetermination" (Stegmüller 1986: 29 und 430, Balzer 1985: 26, zur Anwendung dieser Regel bei den Balancetheorien vgl. Kap. 6.2, S.127 und Kap. 9.3, S.192-195). Unabhängig davon, wie die Extension von I bestimmt wird, kann jede erfolgreiche Ausdehnung der Menge intendierter Anwendungen als *empirischer Fortschritt* betrachtet werden (Stegmüller 1986: 114).

Zwei Eigenschaften von I sind besonders hervorzuheben (Stegmüller 1986: 28).

- Die intendierten Anwendungen einer Theorie sind *unabhängig* von der mathematischen Struktur gegeben und werden nicht mit dieser automatisch mitgeliefert.
- Die Menge I ist eine *offene Menge*, die im historischen Verlauf in der Regel größer, aber auch - bei hartnäckigem Versagen einer Theorie - kleiner werden kann.

Zwischen der Menge intendierter Anwendungen und dem Theoriekern besteht nun folgende Beziehung. Das mindeste, was von der Menge I erwartet werden muß ist, daß diese in der Begrifflichkeit der Theorie darstellbar sein muß, und zwar jener Begrifflichkeit, in der die theoretischen Größen nicht vorkommen. Es muß also gelten: $I \subseteq M_{pp}$.

Mit der Forderung $I \subseteq M_{pp}$ wird ein Aspekt der *Theorienbeladenheit empirischer Beobachtungen* ausgezeichnet. Denn um eine Entität als intendierte Anwendung einer Theorie auszuwählen, muß diese in der Begrifflichkeit der Theorie und damit "durch die Brille der Theorie" betrachtet werden. "Diese theoriegeleitete Strukturierung der Realität erfolgt dadurch, daß ganz bestimmte Objektklassifikationen gewählt werden und daß nur ganz bestimmte Beziehungen zwischen diesen Objekten identifiziert und benannt werden" (Westermann 1987: 30). Die intendierten Anwendungen, also die Elemente der Menge I, sind dabei nicht als reale Systeme zu verstehen, sondern als bereits sprachliche, mit bestimmten Begriffen erfaßte Systeme (Balzer 1985: 26).

Empirische Wissenschaftler wollen aber nicht nur behaupten, daß sich die intendierten Anwendungen in der Begrifflichkeit der Theorie darstellen lassen. Vielmehr soll die viel engere Relation gelten, daß *alle Axiome* - insbesondere das Fundamentalgesetz - auf die Menge I zutreffen und somit alle intendierten Anwendungen auch *Modelle* sind, d.h. es muß die viel stärkere Forderung gelten:
$I \subseteq M$.

Man nennt $I \subseteq M$ die *empirische Behauptung einer Theorie T* (Balzer 1982: 31).[50]

Die empirische Behauptung kann zunächst wahr oder falsch sein. Dadurch, daß die Menge I aber weder intensional noch extensional streng festgelegt ist, kann man jedoch immer beschließen, bestimmte Erweiterungen der Menge I_0 zurückzunehmen. Wenn ein versuchsweises $i \in I$ kein Modell ist, so wird nicht die Theorie bzw. der Theoriekern "falsifiziert", sondern der Theoretiker kann einfach sagen: "i ist keine intendierte Anwendung meiner Theorie" und i einfach aus der Menge I streichen. Stegmüller (1986: 115) spricht in diesem Fall von *empirischem Rückschritt.*

Mit der grundsätzlichen Offenheit der Menge I liegt ein zweiter und für uns wichtigerer Grund für die Immunität von Theorien gegen Widerlegung vor. Widerstreitende Daten müssen nicht gegen die Theorie gewertet werden, sondern der betreffende Realitätsausschnitt kann einfach aus dem Anwendungsbereich *ausgeschlossen* werden. Ein von Stegmüller häufig benutztes Beispiel für die Rationalität dieser Reaktion ist wieder die Partikelmechanik. Newton hat für seine Theorie als typische Anwendungen I_0 genannt: das Sonnensystem und Teilsysteme, den freien Fall in der Nähe der Erdoberfläche, Pendelbewegungen und die Gezeiten. Newton hatte darüberhinaus die Hoffnung gehabt, auch Lichtphänomene in die intendierten Anwendungen seiner Theorie einbeziehen zu können. Für eine gewisse Zeitspanne war nicht klar, ob Lichterscheinungen von Newtons Theorie erklärt werden konnten. Als man später diesen Gedanken preisgab und die Maxwellsche Theorie des Lichts und der Elektrizität akzeptierte, erklärte man deshalb nicht die Theorie von Newton für widerlegt, sondern sagte bloß, daß Licht nicht aus Partikeln besteht (Stegmüller 1979: 487).

Da die eben dargelegte Konsequenz des strukturalistischen Programms in scharfem Konflikt mit den Forderungen der Kritischen Rationalisten steht (Stegmüller 1980: 125), sind einige Bemerkungen zu den unterschiedlichen Auffassungen - aber auch Ähnlichkeiten - beider Metatheorien angebracht. Im Verständnis der Kritischen Rationalisten sind Theorien strengen Falsifikationsversuchen auszusetzen und gegenüber potentieller Widerlegung so empfindlich wie möglich zu machen. Nach dem Popperschen Konzept müßten insbesondere notwendige und hinreichende Bedingungen für die Zugehörigkeit zur Menge intendierter Anwendungen einer Theorie scharf definiert sein. Genau dies bestreitet der Strukturalismus unter Hinweis auf die Wissenschaftsgeschichte. Es scheint nämlich kein Naturwissenschaftler jemals bereit gewesen zu sein, das Falsifikationsrisiko einzugehen, das mit einer expliziten Definition des Umfanges von I, also mit der Angabe notwendiger und hinreichender Bedingungen für die Zugehörigkeit zu I, gegeben wäre. "Gegen diese Enthaltsamkeit von Naturforschern ankämpfen zu wollen, hieße nicht, diese Tätigkeit rationaler zu

[50] Dies ist eine sehr grobe Vereinfachung. In dieser Formulierung trifft die empirische Behauptung nur für Theorien *ohne* theoretische Terme zu. In der strukturalistischen Literatur wird die empirische Behauptung allgemeiner und viel komplizierter unter Verwendung des sog. Ramsey-Satzes formuliert. In dieser Fassung ist die empirische Behauptung ein Existenzsatz: es wird die Existenz "passender" theoretischer Terme behauptet, so daß sich die Elemente von I auf solche Weise (theoretisch) ergänzen lassen, daß die Resultate dieser Ergänzung Modelle sind. Da bei den hier behandelten Theorien theoretische Terme keine Rolle spielen, verzichten wir in diesem Punkt auf die allgemeine Darstellung und verweisen den interessierten Leser auf die wissenschaftstheoretische Literatur.

machen, sondern würde nur den Versuch darstellen, die Vorgänge in der Wissenschaft nach einem vorgefaßten und überspannten Rationalitätsklischee zurechtzubiegen" (Stegmüller 1980: 125-126).

Der Strukturalismus entwirft ein realistischeres Bild empirischer Forschung, das der Kritik von Kuhn an Popper Rechnung trägt. Empirische Untersuchungen werden in der Regel durchgeführt, um die Anwendbarkeit einer Theorie zu zeigen und nicht, um diese zu widerlegen. Wenn eine Theorie in der Vergangenheit gute Dienste geleistet hat, wird man sie nicht preisgeben, nur weil ein oder mehrere Wissenschaftler beim Umgang mit dem Kern keinen Erfolg haben. "Scheitert der Wissenschaftler mit seinen Bemühungen, eine bestimmte Theorie auf bestimmte Arten von Partialmodellen erfolgreich anzuwenden, gibt es nach der strukturalistischen Theorienkonzeption keinen Grund, dieses Scheitern der Theorie in dem Sinn anzulasten, daß man sie als 'falsifiziert', 'belastet' oder dergleichen bezeichnet. Wenn man schon mit den Begriffen rational und irrational arbeiten will, so ist es aus strukturalistischer Sicht ganz und gar irrational, wenn ein Wissenschaftler eine Theorie verwirft, weil er bei ihrer Anwendung in bestimmten Kontexten erfolglos blieb, obwohl die Theorie sich doch zumindest bei den Elementen der paradigmatischen Anwendungsmenge I_0 als erfolgreich erwiesen hat und vielleicht auch noch Generationen von Wissenschaftlern gute Dienste leisten wird" (Westermann 1987: 79).

Vieles von dem, was unter einem orthodoxen Falsifikationismus als methodologisch bedenklich oder einem "dogmatischen Geist" entsprungen scheint, findet aus strukturalistischer Sicht ganz natürliche Erklärungen und verliert den Anspruch des Irrationalen (Stephan 1990: 3). Dies läßt sich auch an sozialwissenschaftlichen Theorien, wie den im zweiten Teil behandelten Balancetheorien belegen. Anderson (1979: 456) stellt die provokante Frage, warum Balancetheorien empirisch so erfolgreich waren, obgleich Gegenbeispiele zu deren Behauptungen leicht zu finden sind. Dieser für einen Falsifikationisten paradoxe und erschütternde Tatbestand hat für den Strukturalisten eine einfache und intuitive Lösung: die Balancetheorien waren erfolgreich, weil sie eine bestimmte Menge empirischer Systeme gut erklären können und die Gegenbeispiele schlicht keine Anwendungen der Theorie sind. Die strukturalistische Auffassung entspricht eher den Intuitionen und dem tatsächlichen Verhalten von Sozialwissenschaftlern. Hallinan (1974) berichtet z.B. von Daten, welche den Voraussagen einer Version der Balancetheorie widersprechen und führt dies auf eine unangemessene Anwendung der Theorie zurück: "I have found that an inappropriate application of the theory is responsible for the unsuccessful predictions of the model" (Hallinan 1974: 365).

Die strukturalistische Theorienkonzeption und die Falsifikationsmethodologie Poppers müssen dennoch nicht als grundsätzlich inkommensurabel betrachtet werden. Popper (1982: 6) versteht seine Wissenschaftsphilosophie in erster Linie als Methode der systematischen Überprüfung von Hypothesen und Theorien und weniger als "Wissenschaftsarchitektur", wie sich die strukturalistische Wissenschaftstheorie darstellt (Balzer/Moulines/Sneed 1987). Westermann (1987: 154-157) sieht den Strukturalismus als generelle Metatheorie, in die sich die deduktive Methodologie Poppers insofern einordnen kann, als sie eine wertvolle Analyse der Methoden der

systematischen Überprüfung von Hypothesen und Theorien bereitstellt. In der strukturalistischen Konzeption sind Theorien zwar keine falsifizierbaren Entitäten, trotzdem können wir aber die methodologische Regel akzeptieren, daß stets kritisch und streng (aber auch genügend wohlwollend) zu prüfen ist, ob eine bestimmte Theorie auf ein bestimmtes empirisches System erfolgreich anwendbar ist. Dies ist in den Sozialwissenschaften um so wichtiger, als bei Anwendung der strukturalistischen Metatheorie auf sozialwissenschaftliche Phänomene eine erhebliche Mißbrauchsgefahr besteht. "Ein grundsätzliches Mißverständnis und ein grober Mißbrauch des Strukturalismus läge insbesondere vor, wenn die von ihm betonte Nichtfalsifizierbarkeit von Theorien als Rechtfertigung für eine beliebige, sich an keinen erkenntnistheoretischen oder methodologischen Überlegungen orientierende empirische Forschung herhalten müßte" (Westermann 1987: 153). Das Poppersche Falsifikationskonzept stellt zweifellos ein wertvolles Instrumentarium bereit, das hilft zu entscheiden, ob der Theoriekern auf bestimmte empirische Systeme erfolgreich angewendet werden kann oder nicht.

Mit der Menge I der intendierten Anwendungen sind die wichtigsten Komponenten des strukturalistischen Programms für unsere Zwecke nun vollständig charakterisiert: eine empirische Theorie besteht zusammenfassend also erstens aus formal eindeutig definierten Mengen von Modellen, potentiellen Modellen und Partialmodellen - dem Theoriekern - und zweitens aus der davon unabhängig und pragmatisch festgelegten Menge der intendierten Anwendungen. Eine empirische Theorie T ist dann das Tupel $T = \langle M, M_p, M_{pp}, I \rangle$.

Wir haben des strukturalistischen Theorienkonzept nur kurz, sehr grob und ohne detaillierte Beispiele skizziert. Auf wichtige Grundbegriffe wie Constraints oder eine ausführlichere Darstellung theoretischer Begriffe wurde verzichtet, da diese in unserem Rahmen nicht von Bedeutung sein werden. Einige pragmatische Begriffe werden in Kap. 12 noch nachgeholt. Eine tiefergehende Beschreibung mit weiterführender Literatur findet sich in Stegmüller (1980, 1986), Balzer (1982) und vor allem Balzer/Moulines/Sneed (1987). Soziologische und sozialpsychologische Beispiele, an denen die eingeführten Begriffe verdeutlicht werden, folgen im zweiten Teil. Wir wollen die kurze Skizzierung des strukturalistischen Theorienkonzepts mit einigen Anmerkungen zur Beziehung zwischen der strukturalistischen Wissenschaftstheorie und den Sozialwissenschaften bzw. der Computermodellierung abschließen.

5.3 Zum Verhältnis strukturalistische Theorienkonzeption - Sozialwissenschaft

In vielen Disziplinen wie Ökonomie oder Psychologie hat die strukturalistische Theoriensicht eine wohlwollende Beachtung erfahren. In der Psychologie findet sich z.B. eine Reihe von Substanzwissenschaftlern, die sich mit diesem Ansatz in kritischer und konstruktiver Weise auseinandersetzen (vgl. z.B. Westmeyer 1992, Westermann 1987, Stephan 1990 und die dort gegebenen Literaturhinweise). Ein herausragendes Beispiel ist die Rekonstruktion der Dissonanztheorie von Festinger (1978, zuerst 1957), die von Westermann (1987) vorliegt und auf dessen Arbeit wir uns bei der

allgemeinen Darstellung öfters bezogen haben. Westermann (1987: 12) nennt fünf Gründe, warum er für die Präzisierung der Dissonanztheorie den Strukturalismus gewählt hat. Erstens erscheint ihm dieser Ansatz in der gegenwärtigen Fachdiskussion der am stärksten bevorzugte zu sein. Zweitens wird er - im Gegensatz zur Carnap- und Popper-Schule - auf weite Bereiche nicht-physikalischer Theorien angewendet. Drittens hat der Strukturalismus in der Psychologie schon früh eine wohlwollende Rezeption erfahren. Viertens erscheint ihm der Strukturalismus dadurch, daß für die Theorien keine vollständige Formalisierung erforderlich ist, in mehrfacher Hinsicht besser geeignet. Fünftens schließlich lassen sich in seinem Rahmen auch andere wissenschaftstheoretische Ansätze gut einordnen.

Im Gegensatz zu seiner relativen Popularität in der Psychologie ist das strukturalistische Programm in der Soziologie bislang auf geringe Resonanz gestoßen. Während wissenschaftstheoretisch interessierte Soziologen das Werk Stegmüllers z.T. detailliert verarbeiten, wird die "strukturalistische Wende" nicht zur Kenntnis genommen bzw. für die Sozialwissenschaften als irrelevant erachtet. In kaum einem Grundlagenlehrbuch findet sich ein Hinweis auf diesen neueren Ansatz geschweige denn eine breitere Auseinandersetzung damit. Schnell/Hill/Esser (1993: 114) erwähnen beispielsweise im ausführlichen Wissenschaftstheorie-Teil ihrer Methodeneinführung den Strukturalismus nur in einer Fußnote mit dem Hinweis, daß dieser in der Methodologie der Sozialwissenschaften "(noch) keine Rolle" spielt. Eine ähnliche Bemerkung findet sich in einem - allerdings bereits älteren - Lehrbuch von Opp (1976: 75), wonach der Strukturalismus "für die meisten Sozialwissenschaften zumindest zum gegenwärtigen Zeitpunkt kaum von Interesse sein dürfte". Leider geben weder Schnell/Hill/Esser noch Opp eine Begründung für ihre These.

Ein vernünftiger Grund, warum die strukturalistische Theorienkonzeption zwar auf psychologische, nicht aber auf soziologische Theorien in fruchtbarer Weise anwendbar sein soll, scheint uns aber nicht gegeben. Beide Disziplinen sind sowohl inhaltlich als auch forschungslogisch eng miteinander verbunden. In ähnlicher Weise wie Westermann sehen wir folgende Stärken des Sneedschen Programms. Erstens bietet es eine einheitliche, liberale Metatheorie für geistes-, sozial- und naturwissenschaftliche Theorien an, ohne sich wie die Vorläufer einseitig an naturwissenschaftlichen Präzisionsvorstellungen und Beispielen zu orientieren. Zweitens stellt es einen Rekonstruktionsapparat ohne übermäßig viel Formalisierungsaufwand in einem geschlossenen formalen Rahmen bereit. Die benötigten formalen Kenntnisse sind im Vergleich zu früheren Ansätzen als verhältnismäßig gering anzusetzen. Drittens erlaubt der Strukturalismus einen relativ unkomplizierten, aber präzisen Vergleich von Theorien und kann sogar die Entwicklung umfassender Forschungsprogramme sowie pragmatische Aspekte erfassen (vgl. Kap. 12). Viertens schließlich entwirft der strukturalistische Ansatz ein neues und realistischeres Bild von empirischer Forschung als frühere Konzepte.

Das strukturalistische Vorgehen unterscheidet sich wesentlich von den in Kap. 2.7 erwähnten Formalisierungsversuchen in den Sozialwissenschaften, in denen versucht wird, qualitative Begriffe zu quantifizieren und in ein höheres Skalenniveau zu transferieren oder ursprünglich qualitative Zusammenhänge in quantitative zu überführen.

Bei der Rekonstruktion der Dissonanztheorie von Festinger erfolgen z.B. keine künstlichen Präzisierungen: "Sowohl der Typ der Begriffe als auch die Art der Zusammenhangshypothesen werden aus der verbalen Formulierung übernommen: Klassifikatorische Begriffe beispielsweise bleiben klassifikatorische Begriffe und werden nicht etwa in kontinuierliche metrische Variablen 'übersetzt'; verbale Aussagen über einen monotonen Zusammenhang zweier Größen werden - um ein zweites Beispiel zu geben - auch in relationstheoretischer Schreibweise Aussagen über einen monotonen Zusammenhang bleiben und nicht etwa durch Angabe einer bestimmten Funktionsregel (wie einer bestimmten Potenzfunktion) 'präzisiert'" (Westermann 1987: 21).

Balzer (1982: 277) verweist darauf, daß die Charakterisierung von Modellen im Strukturalismus so schwach ist, daß sie keinen Angriffspunkt bietet für Unmöglichkeitsargumente der Axiomatisierung, wie sie oft in den Sozialwissenschaften vorgebracht werden. Die einzige echte Forderung ist, daß mit einer Theorie eine bestimmte Anzahl von Grundbegriffen gegeben ist und daß die Relationen einen festen Typ haben. In den Sozialwissenschaften findet sich freilich eine Vielzahl von Beispielen, in denen Begriffe und Relationen inkonsistent verwendet werden. Beispielsweise wird eine Relation oft als 1-stellige und im gleichen Kontext als 2- oder 3-stellige Relation gebraucht. Bei strukturalistischen Rekonstruktionen muß genau spezifiziert werden, "ob es sich bei einem verwendeten Begriff um die Bezeichnung einer einfachen Menge, einer Relation oder einer Funktion handelt. Bei Relationsbegriffen muß dann angegeben werden, auf welche Menge von Grundbegriffen sie sich beziehen und von welchem Typ sie sind" (Westermann 1987: 21). In analoger Weise wie die Computermodellierung übt der Strukturalismus somit einen heilsamen Zwang auf den Theoretiker aus, indem er eine einheitliche Verwendung der Grundbegriffe und Relationen begünstigt. Während in Computerprogrammen aber gewöhnlich zwischen Definitionen, Gesetzen, Axiomen und Theoremen nicht unterschieden wird, liefert eine konkrete strukturalistische Rekonstruktion eine scharfe Unterscheidung zwischen diesen Entitäten. Dies ist insofern bedeutsam, als in den Sozialwissenschaften oft nicht klar ist, ob eine Aussage definitorischen oder gesetzesartigen Charakter hat (wie Opp 1976: 195-196 z.B. an der Verwendung des Rollenbegriffs verdeutlicht).

5.4 Strukturalistisches Theorienkonzept und Computermodelle

Abschließend zu diesem wissenschaftstheoretischen Kapitel sollen einige Parallelen zwischen der strukturalistischen Theorieninterpretation und der KI-Modellierung aufgezeigt werden, und es soll das Verhältnis zwischen Theorien und Computerprogrammen aus strukturalistischer Sicht geklärt werden.

Der wissenschaftstheoretische Strukturalismus und die Expertensystemtechnik sind metatheoretische Konzepte: sowohl die Expertensystemtechnik als auch der Strukturalismus beschäftigen sich mit der Formalisierung von Wissen. Der Strukturalismus kann als eine Technik oder ein Ansatz zur "formalisierten informellen" Darstellung einer besonderen, hervorgehobenen Klasse menschlichen Wissens betrachtet werden,

nämlich empirischen Theorien. Die Wissensrepräsentations- und Expertensystem-Formalismen sind ebenfalls Techniken zur "formalisierten informellen" Darstellung des Wissens von Experten oder allgemeiner, von menschlichen Wissensträgern. Beide Ansätze können als Metatheorien interpretiert werden, und beide Metatheorien vertreten ein liberaleres Konzept im Vergleich zu ihren Vorläufern: Die strukturalistische Wissenschaftstheorie nimmt Abstand von der streng normativen und physikalistischen Orientierung ihrer Vorgänger. Die Wissensverarbeitung erlöst den Modellbauer von der entschieden algorithmisch und numerisch ausgerichteten Modellierung konventioneller Ansätze. "Beide Metatheorien versuchen, einen Kompromiß zu schließen zwischen einer streng exakten Formulierung von Theorien und einer rein pragmatischen, 'natürlichsprachlichen' Repräsentation. Begriffsgerüste, theoretische Terme und Axiome werden nicht exakt in einem Logikkalkül formuliert, eine Modellbildung erfolgt vielmehr durch intuitive Prädikation unter Zuhilfenahme einer 'informellen' Mengenlehre" (Bauer 1989: 3).

Da wissensbasierte Systeme sich zur Darstellung insbesondere nicht-quantifizierbarer Theorien eignen und der Strukturalismus eine Metatheorie zur informellen Rekonstruktion von Theorien ist, in denen Begriffe und Zusammenhänge nicht notwendigerweise quantifiziert werden müssen, können beide Ansätze in fruchtbringender Weise miteinander verknüpft und kombiniert werden und ergänzen sich. "Die Expertensystemtechnik bietet z.B. verschiedene, auch sehr globale Ansätze zur formalisierten 'informellen' Darstellung von Wissen. Wie die Wissenschaftstheorie versucht sie teilweise sehr vage formulierte wissenschaftliche Theorien (Expertenwissen) zu präzisieren, zu formalisieren und zu prozeduralisieren, d.h. nicht nur das Wissen an sich, sondern auch den Prozeß der Wissensakquisition, den Ableitungsprozeß und den Reflexionsprozeß des Wissenschaftlers (Experten) zu fixieren. Aufgrund der verwandten Ziele beider Gebiete scheint es vernünftig, die Erkenntnisse und die Hilfsmittel des jeweils Anderen zu nutzen" (Bauer 1989: 1-2).

Die Parallelen gelten insbesondere auch auf einer unteren Ebene für den Strukturalismus und PROLOG. PROLOG beschreibt Probleme in Form logischer Axiome. Der Strukturalismus rekonstruiert Theorien axiomatisch mengentheoretisch. Damit sind beide Formalismen mächtige Ergänzungen, denn die Axiome des Strukturalisten lassen sich z.B. in PROLOG darstellen.

Wie kann nun ein Computerprogramm, das eine Theorie repräsentiert, aus Sicht der strukturalistischen Wissenschaftstheorie interpretiert werden? Nach den allgemeinen Ausführungen ist die Korrespondenz leicht herzustellen.

Grundsätzlich sind sowohl Computerprogramme als auch Axiome sprachliche Einheiten, und beide haben die Aufgabe, andere Einheiten zu charakterisieren und exakter festzulegen. Programme legen allgemein Programmabläufe fest, Axiome allgemein die genaue Form der Strukturen. Der Menge der Programmabläufe entspricht also die Menge der von den Axiomen festgelegten Modelle. Dem konkreten Programm - d.h. der Menge von Regeln - auf der Computerseite entsprechen auf der Modellseite somit exakt die Axiome für eine Theorie, welche eine Modellklasse definieren.

Zwischen einem konkreten Programmlauf und einem konkreten Modell besteht ebenfalls eine Entsprechung. Ein bestimmtes Modell enthält bestimmte, konkrete

Objekte und Relationen, ein Programmablauf verändert einen bestimmten, konkreten Input und überführt ihn in einen konkreten Output. Ein faktischer Programmablauf entspricht somit einem faktischen Modell. Da die Modellseite allgemein gehalten ist, haben Input, Output und Transformation von Input in Output auf Computerseite keine Gegenstücke. Input, Output und Transformation bilden jedoch Teile von Modellen, wenn wir Modelle spezialisieren. Je nach Herkunft der Datenstruktur kann diese als Repräsentant eines realen oder fiktiven intendierten Systems betrachtet werden.

Auf die oben angeschnittene Frage, inwieweit Programme als Theorien zu betrachten sind, hat die strukturalistische Wissenschaftstheorie also eine eindeutige Antwort: Computerprogramme sind *keine* Theorien. Programme entsprechen den Axiomen für eine Theorie. Der Klasse der Modelle entspricht auf Computerseite die Menge aller Programmabläufe, und ein konkretes Modell entspricht einem Programmablauf.

In dieser Interpretation ist der Begriff des Computermodells metaphorisch zu verstehen. Analog wie Axiome bestimmte Strukturen als Modelle auszeichnen, kann ein Computerprogramm bestimmte Abläufe festlegen und damit "modellieren". Nicht Axiome oder Computerprogramme sind Modelle, sondern beide charakterisieren nur bestimmte Phänomene und legen sie fest. "Computermodell" kann somit einfach als Metapher für "Computerprogramm zur Modellierierung empirischer Phänomene" verstanden werden.

Teil II

Balancetheorien: Logische Struktur und wissensbasierte Modellierung

6. Die Balancetheorie von Heider

Die im ersten Teil gemachten allgemeinen Ausführungen sollen nun konkret auf Balancetheorien angewendet werden. Die Plattform des zweiten Teils bildet dabei die Beschreibung einer Folge von Balancetheorien in ihrer historischen Entwicklung. Eine Teilmenge dieser Theorienfolge werden wir im strukturalistischen Format rekonstruieren und in regelbasierte Modelle umsetzen. Am Schluß soll das ganze balancetheoretische Programm einer wissenschaftstheoretischen Analyse unterzogen werden. Wir geben zunächst einen kurzen Überblick über die hier verfolgte theoretische Entwicklungslinie.

Ausgangspunkt und Basis des balancetheoretischen Programms ist das sozialpsychologische Modell von Fritz Heider (1946, 1977). Es postuliert, daß Menschen nach Konsistenz streben und versuchen, zu anderen Personen und Objekten ausgewogene Beziehungen herzustellen. Heider betrachtet einfache kognitive Individualsysteme. Eine erste Generalisierung ist das Modell von Abelson/Rosenberg (1958), das bestimmte Beschränkungen bei Heider aufhebt, aber immer noch die kognitive Sicht beibehält. Cartwright/Harary (1956) verallgemeinern das Balancemodell graphentheoretisch und wenden es erstmals auf "objektiv beobachtbare" soziale Netze an. Dabei zeigen die Autoren mit dem Strukturtheorem die makrostrukturellen Auswirkungen auf das Gesamtsystem als Folge von Balance auf.

Mit den D-H-L-Modellen rücken die makrostrukturellen Konsequenzen von Gleichgewicht vollends in den Mittelpunkt. Anknüpfend an das verallgemeinerte Heider-Modell werden logische Zusammenhänge zwischen der Balancestruktur auf individueller Ebene und der Struktur auf Makro-Ebene untersucht. Die D-H-L-Modelle - benannt nach den Urhebern James A. Davis, Paul W. Holland und Samuel Leinhardt - zeigen, daß balancierte Strukturen auf Mikro-Ebene, die bestimmten Bedingungen genügen, makrostrukturell eine horizontale Vercliquung (Davis 1967) und vertikale Hierarchisierung (Davis/Leinhardt 1972) zur Folge haben. Schließlich wird mit dem Konzept des transitiven Graphen ein noch allgemeineres Modell eingeführt, welches die vorher genannten Versionen als Spezialfälle enthält.

Die ursprüngliche Balancetheorie in der Fassung von Heider hat in den fünfziger bis siebziger Jahren eine entscheidende Initialisierungsrolle in den Sozialwissenschaften gespielt. Zum einen hat sie eine große Zahl von Experimenten angeregt und die Formulierung anderer - komplexerer - Gleichgewichtstheorien initiiert, zum andern hat sie

die Anregung gegeben, genauer über Formalisierungsmöglichkeiten in den Sozialwissenschaften nachzudenken (Sukale 1971: 48). In Bezug auf unser Vorhaben der logischen Rekonstruktion und regelbasierten Modellierung erscheinen Balancetheorien aus zwei Gründen ideal. Erstens ist die theoretische Entwicklung relativ überschaubar und abgeschlossen. Zweitens sind diese Theorien strukturell ausgerichtet und damit prädestinierte Kandidaten für einen wissensbasierten Modellierungsansatz.

Wir gehen im folgenden so vor, daß wir das Ausgangsmodell von Heider zunächst informell vorstellen und anschließend seine logische Struktur in einer präzisierten, strukturalistischen Deutung. An diesem sehr einfachen Beispiel wird das oben nur formlos skizzierte strukturalistische Instrumentarium und Vorgehen nochmals erläutert. Die PROLOG-Implementierung stellen wir in zwei Varianten vor: einmal als Datengenerierungs-Modell und zum andern als Computerprogramm, in dem ausschließlich strukturalistische Prädikate definiert werden. Im weiteren werden die strukturalistischen Prädikate gleich innerhalb der informellen Darstellung entwickelt und die Programme ausschließlich als Datengenerierungs-Modelle.

6.1 Informelle Darstellung[51]

Die Wurzeln der Theorie von Heider - und damit aller anderen Balancetheorien - liegen zum ersten in Spinozas Ethik, zum zweiten in dem feldtheoretischen Ansatz von Kurt Lewin und zum dritten in der Gestaltpsychologie, die in den dreißiger Jahren von Koffka, Köhler und Wertheimer begründet wurde. Von allen drei Einflußfaktoren wollen wir nur den gestaltpsychologischen kurz andeuten, da er uns als der zentralste erscheint.[52]

Gestaltpsychologen gehen davon aus, daß Menschen nach der "guten Gestalt" oder "guten Form" streben und gewisse Konfigurationen wegen ihrer Einfachheit und Kohärenz bevorzugen. Dieser Wunsch nach der guten Gestalt hat zur Folge, daß Personen versuchen, Inkonsistenzen zu vermeiden und sich um eine geordnete und kohärente Sicht der Welt bemühen. Nach der - schon im Altertum erkannten - Konsistenzregel wird die Verbindung von Einzeleindrücken zu einem Gesamteindruck so vorgenommen, daß kein Widerspruch enthalten ist (Witte 1989: 324).[53] Dem Konsistenzprinzip zufolge trachten Personen im allgemeinen danach, ihre Einstellungen und Überzeugungen (kurz: Kognitionen) untereinander und mit ihrem Verhalten widerspruchsfrei zu organisieren. Beispielsweise sind die beiden Kognitionen "Ich treibe gern Sport" und "Sport ist gesund" konsistent. Hingegen sind die beiden Kognitionen "Ich rauche gern" und "Rauchen ist gesundheitsschädlich" inkonsistent. Inkonsistene Kognitionen erzeugen eine gewisse Spannung und einen psychischen Druck zur Herstellung von Konsistenz: "Nimmt eine Person z.B. an sich Verhaltensweisen wahr,

---

[51] Für die informelle Darstellung der Balancetheorie vgl. Heider (1977), Stahlberg/Frey (1987), West/Wicklund (1985) und Irle (1975).

[52] Die wichtigsten Ideen der Gestaltpsychologie in Zusammenhang mit Sozialpsychologie finden sich in Deutsch/Krauss (1976).

[53] Witte (1989: 325) weist darauf hin, daß bereits Aristoteles das unten von Heider behandelte Prinzip formulierte, wonach die Freunde unserer Freunde auch unsere Freunde sein werden.

die ihren Einstellungen widersprechen, so befindet sie sich in einem Zustand kognitiver Inkonsistenz. Einen solchen Zustand wird sie ... als unangenehm, da Spannung erzeugend, erleben. Sie wird deshalb motiviert sein, die beteiligten Kognitionen (wiederum) in ein konsistentes und damit spannungsfreies Verhältnis zu überführen, indem sie einzelne dieser Kognitionen oder auch ihr Verhalten ändert" (Stahlberg/Frey 1987: 214).

Auf der Basis des Konsistenzprinzips wurden in der Sozialpsychologie verschiedene theoretische Konzepte entwickelt, denen die grundlegende Annahme gemeinsam ist, daß Inkonsistenz Spannung erzeugt und eine Tendenz zur Herstellung von Konsistenz impliziert. Zu den bekanntesten sozialpsychologischen Konsistenztheorien gehört - neben der hier behandelten Balancetheorie - die bereits erwähnte Dissonanztheorie von Festinger (1978).

Die Gleichgewichtstheorie ist eine spezifische Konsistenztheorie. Sie beschäftigt sich mit der Widersprüchlichkeit in den Beziehungen, die eine Person zwischen sich und anderen Elementen ihrer Umwelt wahrnimmt. Ausgangspunkt ist Fritz Heiders Artikel "Attitudes and Cognitive Organization" aus dem Jahr 1946. In diesem Aufsatz macht Heider die grundlegende Annahme, daß soziale Wahrnehmung den eben beschriebenen gestaltähnlichen Strukturprinzipien folgt und *ausgeglichene oder balancierte Zustände gegenüber unausgeglichenen oder unbalancierten Zuständen präferiert* werden. Gleichgewichtige Systeme werden als angenehm und kohärent im Sinne der Gestaltpsychologie empfunden: "Mit Gleichgewichtszustand ist eine Situation gemeint, in der die Relationen zwischen den Größen harmonisch zueinander passen; es gibt keinen Drang zu einer Veränderung" (Heider 1977: 238).

Die Zustände "Ausgeglichenheit" und "Unausgeglichenheit" werden in der Heider-Theorie auf eine kognitive Konfiguration bestehend aus drei Elementen angewendet. Diese Konfiguration wird gebildet aus einer wahrnehmenden Person p, einer anderen Person o und einem impersonalen Objekt x. x kann dabei "eine Situation, ein Ereignis, eine Idee oder irgendein Ding etc." sein (Heider 1946: 107). Genau genommen unterscheidet Heider (1946) noch drei andere Systeme, nämlich dyadische Beziehungen mit zwei Elementen und triadische Konfigurationen mit drei Personen. In Anlehnung an Sukale (1971: 49) wollen wir diese einfacheren Systeme aber außer Betracht lassen.

Ein Beziehungssystem mit den drei Elementen p-o-x bildet eine Triade, die sich grafisch wie in Abb. 6.1 darstellen läßt. Wichtig ist, daß die Situation immer aus der Sicht von p analysiert wird und p, o und x dabei als Einheiten in der kognitiven Struktur von p repräsentiert sind.

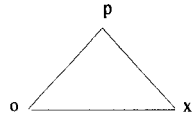


Abb 6.1: Grafische Darstellung von Triaden

Um festzustellen, ob eine bestimmte Triade als balanciert oder unbalanciert wahrgenommen wird, unterteilt Heider die in der Triade bestehenden Relationen in positive und negative Einheits- oder Gefühlsbeziehungen ("unit" oder "sentiment relations").

Einheitsrelationen U bestimmen, ob je zwei Einheiten zusammengehören oder nicht, wie etwa p besitzt x oder p und o sind sich ähnlich. Gefühlsrelationen L beziehen sich auf eine Bewertung von o oder x durch p und sind Sympathie- oder Antipathierelationen, z.B. p schätzt x oder o verabscheut x. Die Identifikation als Einheits- bzw. Gefühlsrelation ist für die Entscheidung, ob ein System als balanciert wahrgenommen wird oder nicht, aber nicht ausschlaggebend. Bestimmend ist nur die Anzahl der beteiligten positiven und negativen Beziehungen.

Die formale Definition von Gleichgewicht und Ungleichgewicht ist einfach: "Eine Triade ist im Gleichgewicht, wenn alle drei Relationen positiv sind oder wenn zwei Relationen negativ sind und eine positiv ist. Ungleichgewicht tritt dann auf, wenn zwei Relationen positiv sind und eine negativ ist" (Heider 1977: 240). Der Fall mit drei negativen Relationen wird von Heider nicht eindeutig entschieden und als mehrdeutig und unbestimmt bezeichnet (Heider 1946: 110; 1977: 240). Wir kommen auf dieses Problem später noch zurück. Abb. 6.2 veranschaulicht alle möglichen Permutationen positiver und negativer Relationen in einer Triade. Von den so entstehenden acht Fällen sind die Triaden (a-d) im Gleichgewicht, die Triaden (e-g) im Ungleichgewicht und (h) ist die indefinite Triade.

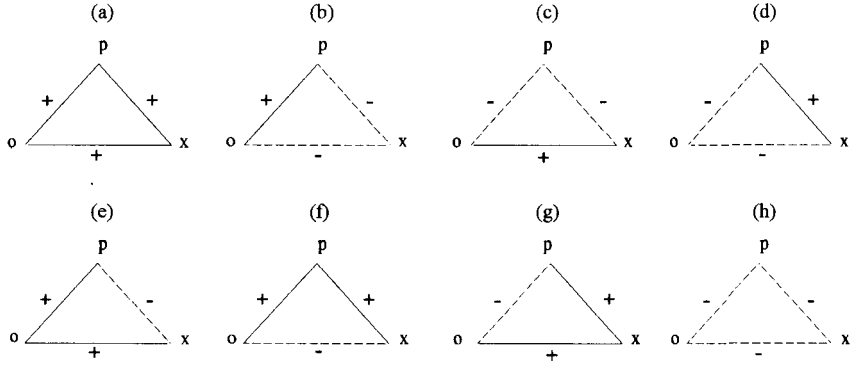


Abb 6.2: Balancierte und unbalancierte Triaden nach Heider. Durchgezogene, mit '+' markierte Linien bedeuten positive, gestrichelte, mit '-' markierte Linien, negative Relationen (nach: Witte 1989: 327). Die Triaden (a-d) sind balanciert, die Triaden (e-g) sind unbalanciert und Triade (h) ist "unbestimmt".

Die balancierten Triaden lassen sich metaphorisch aus der Sicht von p (und unter der Erweiterung, daß x ebenfalls eine Person ist) wie folgt interpretieren:

(a) Der "Freund" meines "Freundes" ist mein "Freund".

(b) Der "Feind" meines "Freundes" ist mein "Feind".

(c) Der "Freund" meines "Feindes" ist mein "Feind".

(d) Der "Feind" meines "Feindes" ist mein "Freund".

Die folgenden Situationen wären Beispiele für balancierte und unbalancierte Triaden (vgl. Heider 1977: 212 und 242):

(1) p ist mit o befreundet und p wählt eine Partei x, die o ablehnt.

(2) zwei Personen p und o sind befreundet und haben die gleiche Einstellung zur Kirche x.
(3) Sie (p) haßt immer die Personen (o), mit denen sie arbeiten (x) muß.
(4) John (p) billigt einen Zug (x) seines politischen Gegners (o).

Konfiguration (1) korrespondiert dabei der unbalancierten Triade (f), (2) der balancierten Triade (a) oder (b), je nachdem, ob p und o eine beidseitig positive oder beidseitig negative Einstellung zur Kirche haben und (3) und (4) der unbalancierten Triade (g). Heider legt eine umfassende Beispielsammlung balancierter und unbalancierter Situationen aus Literatur und Alltagsleben vor.

Das Balanceprinzip von Heider läßt sich nun leicht angeben. Da gleichgewichtige Systeme als angenehm und harmonisch empfunden werden, sind diese über die Zeit stabil und werden nicht verändert. Unbalancierte Systeme werden hingegen als disharmonisch und spannungsvoll empfunden, sind instabil und tendieren zu einer Änderung in gleichgewichtige Systeme: "If no balanced state exists, then forces towards this state will arise ... If a change is not possible, the state of imbalance will produce tension" (Heider 1947: 107-108). *Heiders fundamentale theoretische Aussage besteht also darin zu behaupten, daß ungleichgewichtige Zustände einen psychischen Druck erzeugen und dazu neigen, gleichgewichtig zu werden.* Dieses einfache Gesetz der Gleichgewichtstendenz bildet das grundlegende Prinzip, welches allen Versionen von Balancetheorien gemeinsam ist.

Die Herstellung von Gleichgewicht geschieht im einfachsten Fall durch Änderung der Relationen. In dem Beispiel (1) könnte ein Gleichgewichtszustand dadurch erzeugt werden, daß p versucht, o zur Wahl einer anderen Partei zu bewegen. Eine andere Möglichkeit wäre, daß p seinerseits seine Parteienpräferenz überdenkt. Eine dritte Möglichkeit ist, daß p seine Relation zu o ändert. Die Balancetheorie macht aber keine Vorhersagen darüber, *welche* Veränderungen in einem unbalancierten System zu erwarten sind. Sie behauptet nur, *daß* unbalancierte Triaden dazu tendieren, gleichgewichtig zu werden; wie dies bewerkstelligt wird, bleibt unspezifiziert.

Eine Vielzahl von Experimenten wurde durchgeführt zur Überprüfung der balancetheoretischen Annahmen; wir wollen auf einige davon hinweisen.

Grundsätzlich können sich empirische Untersuchungen auf zwei Aspekte der Behauptung von Heider stützen. Man kann eine Triade nach dem Grad des gefühlsmäßigen Wohlbefindens beurteilen oder nach ihrer Stabilität. Beide Perspektiven sind eine Konsequenz der theoretischen Annahme, daß balancierte Zustände als angenehm empfunden werden und stabil bleiben. In der Literatur stützt man sich in der Regel auf die erste Möglichkeit. Beispielsweise ließ Jordan (1953) 64 hypothetische p-o-x Triaden nach dem Grad ihres Wohlbefindens einstufen. Es zeigte sich wie erwartet, daß die vier balancierten Triaden signifikant positiver bewertet wurden als die unbalancierten. Allerdings ergaben sich auch Unterschiede *innerhalb* von balancierten Triaden: die Triaden (a) und (b) von Abb. 6.2 wurden als signifikant angenehmer erlebt als die anderen balancierten Triaden.

Viele Untersuchungen weisen auf unheilvolle Implikationen des Balanceprinzips für das Alltagsleben hin. Als Beispiel zitieren wir ein Experiment von Landy/Aronson (1969). In diesem Experiment mußten Studenten (p) die Rolle von Geschworenen

übernehmen, die einen Angeklagten (o) wegen einer schweren Straftat (x) verurteilen sollten. Experimentell variiert wurde die Attraktivität des Angeklagten (attraktiv - neutral - unattraktiv). Nach der Balancetheorie müßten die Geschworenen bei einem sympathischen Angeklagten dahin tendieren, diesen in jeder Hinsicht positiv zu sehen und also seine Tat milde beurteilen. Bei einem unsympathischen Angeklagten würde Gleichgewicht hingegen hergestellt werden, indem der Angeklagte negativ gesehen wird und eine entsprechend schwere Strafe erhält. Die Ergebnisse entsprachen der Theorie: sympathische Angeklagte wurden zu durchschnittlich 5.5, unsympathische zu 7.1 Jahren Haft verurteilt.

In der Untersuchung von Landy/Aronson wurde ein unsympathischer Angeklagter infolge seiner mangelnden Attraktivität streng bestraft. In einer anderen Studie von Lerner/Simmons (1966) wurde umgekehrt eine Frau, die eine strenge Strafe erhielt, infolge der Strafe als unsympathisch wahrgenommen. Die gleiche Frau wurde hingegen als attraktiv angesehen, wenn sie nicht weiter bestraft wurde. Die Autoren erklären ihr Ergebnis balancetheoretisch damit, daß eine attraktive Frau, die bestraft wird (Elektroschocks erhält), ein Ungleichgewicht erzeugt und der Gleichgewichtszustand durch Umpolung der Attraktivitätskognition wieder hergestellt wird. Beide Untersuchungen lassen auf die Neigung von Menschen schließen, andere entweder durchgängig mit positiven oder durchgängig mit negativen Eigenschaften zu versehen: "Haben wir von jemandem im großen und ganzen eine gute Meinung, sind wir geneigt, Verhalten und Eigenschaften dieses Menschen insgesamt in einem positiven Licht zu sehen. Sind wir jemandem dagegen nicht wohlgesonnen, werden wir wahrscheinlich auch Eigenschaften, die wir neu an ihm entdecken, in diese Einstellung einbeziehen" (West/Wicklund 1985: 157).

Weitere Untersuchungen werden angegeben z.B. in West/Wicklund (1985) und Witte (1989). Das Modell hat sich in einer Vielzahl von Experimenten weitgehend bestätigt und sich bei der Erklärung einer Reihe von Phänomenen wie interpersoneller Anziehung bewährt. Ein dem Gleichgewicht ähnlicher Begriff wurde von Newcomb (1953) mit der A-B-X-Theorie vorgestellt und auf die Kommunikation zwischen Menschen angewandt. Das Kongruenzprinzip von Osgood/Tannenbaum (1955) ist ebenfalls eine frühe Variante der Heider-Theorie, die auf die Richtung von Einstellungsänderungen abzielt.

6.2 Logische Struktur der Heider-Theorie

Wir halten uns bei der Darstellung der logischen Struktur der Theorie genau an die Version, wie sie Heider ursprünglich vorgelegt hat und wie sie eben zusammengefaßt informell vorgestellt wurde. Wie in Kap. 5 dargelegt, wird in der strukturalistischen Rekonstruktion die präsystematisch gegebene Theorie informell mengentheoretisch expliziert und axiomatisch dargestellt. Wenn wir die Theorie in dieser Weise rekonstruieren, müssen wir zunächst überlegen, welche Begriffe und Relationen in der Theorie verwendet werden und von welchem Typ sie sind. In der Heider-Theorie sind die Grundbegriffe und Relationen besonders einfach, letztere aber - wie sich gezeigt

hat - wenig präzise. Heider unterscheidet zwischen zwei Arten von Entitäten, die er als Grundbegriffe behandelt: Personen und Nicht-Personen. Nicht-Personen können dabei eine Vielzahl von Entitäten sein: Ereignisse, Objekte, Situationen etc. Zwischen diesen beiden Grundbegriffen herrschen zwei Typen von Relationen: die gefühlsgeladene L-Relation und die U-Relation. Auf die Problematik der L- und U-Relationen weisen Cartwright/Harary (1956) und Sukale (1971) hin.

Heider formalisiert seine Theorie nicht, sondern symbolisiert sie lediglich, d.h. er verwendet nur abkürzende Symbole zu deren Darstellung. Beispielsweise wird die Gefühlsrelation mit "L" symbolisiert, wenn sie positiv ist und mit "~L", wenn sie negativ ist. Analog wird die Einheitsrelation mit "U" bzw. "~U" symbolisiert.

Unklar bleibt, was das Symbol "~" vor einem Relationssymbol aus der Relation macht: "Das Gegenteil der Relation (z.B. aus Liebe Haß) oder schlicht das Komplement (z.B. aus Liebe Nicht-Liebe, beziehungsweise: die Relation Liebe liegt nicht vor)? Diese irreführende Symbolisierung, die in der mangelhaften Definition des Negationszeichens bestand, machte sich sofort bei den ersten Experimenten bemerkbar, wo Unklarheiten über die Interpretation negierter Relationen auftauchten und zur willkürlichen Klassifikation von Daten führte" (Sukale 1971: 49). Im allgemeinen scheint es so, als ob "~L" in der Bedeutung von "Nichtmögen" - die gegenteilige Relation - aufgefaßt ist, während "~U" als Komplement verwendet wird. In beiden Fällen wird jedoch das gleiche Symbol benutzt.

Diese Unklarheiten wären vermieden worden, wenn L- und U-Beziehungen mit Hilfe der Relationentheorie und allgemeinen Eigenschaften von Relationen beschrieben worden wären, wie es im strukturalistischen Theorienkonzept üblich ist. Da L- und U-Beziehungen im Hinblick auf Balance aber keine Rolle spielen, ersparen wir uns den Umweg über die Codierung in L-und U-Relationen und beziehen uns gleich auf positive und negative Relationen als Grundbeziehungen.

Die in der Heider-Theorie verwendeten Begriffe bestehen also zunächst aus einer dreielementigen Menge O sowie positiven und negativen Relationen. Die erste Definition führt diese Grundbegriffe ein und charakterisiert sie hinsichtlich ihrer mengentheoretischen Typen. Auf der Basis dieser Grundbegriffe können weitere Begriffe definiert werden: die Menge O zerfällt z.B. in die Klasse der Personen und Nicht-Personen und auf der Grundlage der Relationen kann die Menge aller Triaden und Triadenzustände definiert werden.

Die Klasse der Grundbegriffe und Relationen wird normalerweise im potentiellen Modell angegeben. Bevor wir dies tun, müssen wir uns noch einmal die zentrale Behauptung der Balancetheorie verdeutlichen. Das grundlegende inhaltliche Axiom von Heider ist, daß Ungleichgewichtszustände dazu *neigen*, gleichgewichtig zu werden. Wenn wir "neigen" dahingehend interpretieren, daß unbalancierte Strukturen *über die Zeit* hinweg in balancierte überführt werden, dann ist implizit in dieser Behauptung ein Zeitindex enthalten. Damit muß in die Modelldefinition ein Zeitindex eingeführt werden, so daß die Zeit und eine entsprechende Ordnungsrelation auf ihr auch als Grundbegriffe im potentiellen Modell benötigt werden. Andererseits können die obengenannten Grundmengen und Relationen unabhängig von der Zeit definiert werden. Wir führen die grundlegenden Mengen und Relationen deshalb zuerst ohne Bezug-

nahme auf die Zeit in einem Prädikat "...ist ein Heider-Graph"[54] ein und verwenden dieses Prädikat dann zusammen mit dem Zeitbegriff und der auf ihr definierten Ordnungsrelation im Definiens des potentiellen Modells.

Definition 1
x ist ein *Heider-Graph* ($x \in HG$) genau dann, wenn es O, P und N gibt, so daß gilt:
(1) $x = \langle O,P,N \rangle$
(2) O ist eine nicht-leere Menge der Kardinalität 3
(3) $P \subseteq O \times O$
(4) $N \subseteq O \times O$
(5) $P \cap N = \varnothing$
(6) Für alle $x \in O$: $\neg \langle x,x \rangle \in P \cup N$
(7) Wenn $x,y \in O$ dann $\langle x,y \rangle \in P \cup N$ oder $\langle y,x \rangle \in P \cup N$

Definition 1 legt den mengentheoretischen Typ der Grundbegriffe und Relationen fest. Die Axiome besagen, daß O eine dreielementige Menge ist (2) und P und N auf O definierte Relationen (3,4) sind, welche disjunkt (5) und irreflexiv sind (6). In Axiom (7) wird die von Heider implizit getroffene Annahme der Vollständigkeit der Struktur formuliert, d.h. daß zwischen allen Paarelementen aus O entweder eine P- oder N-Relation *existieren muß*. Die uninterpretierten Grundbegriffe sollen folgende intendierte Interpretation haben: O soll eine Menge von Personen und Nicht-Personen sein und P bzw. N die Menge der positiv bzw. negativ verknüpften Objektpaare. $\langle x,y \rangle \in P$ ist zu lesen als: zwischen x und y besteht eine positive Relation und entsprechend $\langle x,y \rangle \in N$: zwischen x und y besteht eine negative Relation.

Auf der Basis dieser Grundbegriffe lassen sich nun verschiedene abgeleitete Beziehungen definieren, die zur weiteren Verwendung nützlich sind. Zunächst wird definiert, was allgemein eine Relation R ist und dann, was eine Nicht-Person (NP) und eine Person (PE) ist (vgl. hierzu auch Sukale 1971: 53).

Definition 2
Wenn $x = \langle O,P,N \rangle \in HG$ dann
(1) $R := P \cup N$
(2) $x \in NP$ genau dann wenn $x \in O$ und es existiert kein y, so daß $y \in O$ und xRy
(3) $x \in PE$ genau dann wenn $x \in O$ und $\neg x \in NP$
(4) $TR := R \times R \times R$

Definition 2-1 legt R als Vereinigung von P und N fest. Die Definitionen 2-2 und 2-3 bestimmen Personen und Nicht-Personen. Personen werden von Nicht-Personen rein formal dadurch unterschieden, daß alles, was kein Anfangspunkt einer Relation ist, eine Nicht-Person ist und Personen diejenigen Elemente, die keine Nicht-Person sind. Definition 2-4 legt die Triadenmenge TR als Menge von Tripeln, also als dreifaches Cartesisches Produkt über R fest.

---

[54] Zum Begriff des Graphen vgl. Kap.8.1.

Als nächstes müssen die balancierten, unbalancierten und indifferenten Triaden von Abb. 6.2 eingeführt werden. Diese werden einfach als Teilmengen I (Indefinit), G (Gleichgewicht) und U (Ungleichgewicht) von TR definiert.

Definition 3
Wenn $x = \langle O,P,N\rangle \in HG$ dann wird TR wie folgt in Teilmengen G, U, $I \subseteq TR$ zerlegt
(1) $I = \{\langle a,b,c\rangle \mid a,b,c \in N\}$
(2) $G = \{\langle a,b,c\rangle \mid a,b,c \in P \vee a \in P \wedge b,c \in N \vee b \in P \wedge a,c \in N \vee c \in P \wedge a,b \in N\}$
(3) $U = \{\langle a,b,c\rangle \mid a \in N \wedge b,c \in P \vee b \in N \wedge a,c \in P \vee c \in N \wedge a,b \in P\}$

Damit können wir nun das potentielle Modell definieren, in dem die Zeit T (genauer: T und die Kleiner-Relation als endliche, lineare Ordnung) als neuer Grundbegriff eingeführt wird und der in Definition 1 festgelegte Heider-Graph auf die Zeit bezogen wird.

Definition 4
x ist ein *potentielles Modell* der Heider-Theorie ($x \in M_p(HT)$) gdw es O, T <, P, N gibt, so daß gilt:
(1) $x = \langle O, T, <, P, N\rangle$
(2) O ist eine dreielementige Menge
(3) $\langle T,<\rangle$ ist eine endliche, lineare Ordnung
(4) $P: T \rightarrow Pot(O \times O)$ und $N: T \rightarrow Pot(O \times O)$
(5) Für alle $t \in T$: $\langle O, P(t),N(t)\rangle \in HG$

Axiom (4) besagt, daß P und N Funktionen sind, die jedem Zeitpunkt $t \in T$ genau ein Element aus der Potenzmenge, also der Menge aller Teilmengen $O \times O$, zuordnen. Axiom (5) fordert, daß das Tripel $\langle O, P(t),N(t)\rangle$ für alle betrachteten Zeitpunkte t ein Heider-Graph ist. In diesem Axiom ist die Forderung enthalten, daß alle Objekte über die verstrichenen Zeiteinheiten die gleichen bleiben müssen. Ansonsten wäre es möglich, zwei ganz verschiedene Grundmengen zu einem Balancesystem zu verbinden, die gar nichts miteinander zu tun haben.

Als nächste Struktur wäre das partielle potentielle Modell zu definieren. Damit ist zu klären, ob theoretische Begriffe in der Heider-Theorie (HT) vorkommen. Diese Frage ist leicht zu entscheiden: es gibt keine Begriffe, zu deren Messung man die Gültigkeit der Heider-Theorie voraussetzen muß. Zwar sind P und N Einstellungen und die Bestimmung von Einstellungen ist wie jede Einstellungsmessung stark theoriegeleitet. Theoretische Begriffe im Sinn der Theorie der Einstellungsmessung gehen aber in HT als nichttheoretische Begriffe ein (Stephan 1990: 75). In der Heider-Theorie existieren somit keine HT-theoretischen Terme, und es besteht keine Notwendigkeit, zwischen partiellen potentiellen Modellen und potentiellen Modellen zu unterscheiden.
Mit Definition 5 ist die Festlegung der Grundbegriffe und abgeleiteten Begriffe abgeschlossen.

Definition 5
$x(t) := \langle O, P(t), N(t) \rangle$

Wir können nun das potentielle Modell durch Hinzufügen des eigentlichen Axioms zu einem Modell ergänzen.

Definition 6
x ist ein *Modell* von HT ($x \in M(HT)$) gdw es O, T, <, P und N gibt, so daß gilt:
(1) $x = \langle O,T,<,P,N \rangle$
(2) $x \in M_p(HT)$
(3) Für alle $t \in T$ und für alle a: Wenn $t < \max(T)$ und $a \in TR_{x(t)}$ und $a \in U_{x(t)}$, dann gibt es ein $t' \in T$ so daß gilt: $t < t'$ und $a \in G_{x(t')}$ und für alle $t'' > t'$: $a \in G_{x(t'')}$

Definition 6 repräsentiert mit Axiom (3) Heiders Kernaussage, nach der unbalancierte Triaden über eine Zeitperiode hinweg in balancierte überführt werden. Definition 6-3 bildet das eigentlich inhaltliche Axiom und Fundamentalgesetz der Balancetheorie. U und G sind dabei die in Definition 3 eingeführten ungleichgewichtigen und gleichgewichtigen Triaden. In unserer Rekonstruktion besagt das Gesetz informell, daß für alle Zeitpunkte t und alle Triaden a gilt: Wenn t kleiner ist als das Maximum von T und a zu t unbalanciert ist, dann gibt es einen Zeitpunkt $t' > t$, bei dem a balanciert ist und für alle Zeitpunkte t", die größer sind als t', bleibt a balanciert. Die folgende Entwicklung der Heider-Triaden über sechs Zeitpunkte würde danach die Modelldefinition erfüllen:

t_1	t_2	t_3	t_4	t_5	t_6
U	U	U	G	G	G

Das zweite Konjunktionsglied nach dem Existenzquantor (für alle $t'' > t'$: $a \in G_{x(t'')}$) ist in unserer Interpretation von Heider nötig. Ansonsten könnte nämlich nach dem erstmaligen Wechsel einer unbalancierten Triade in eine balancierte, diese zu einem späteren Zeitpunkt wieder unbalanciert werden. Das folgende Beispiel zeigt eine solche Situation:

t_1	t_2	t_3	t_4	t_5	t_6
U	U	U	G	U	U

Wir meinen, Heider so zu interpretieren, daß dies ausgeschlossen ist; andernfalls könnte einfach das zweite Konjunktionsglied weggelassen werden.

Das Fundamentalgesetz (6-3) ist *bewußt vage gehalten*, dürfte aber die Intention Heiders genau ausdrücken. Zum einen wird nicht spezifiziert, *welche* Relationen geändert werden, zum andern wird die *Zeitspanne*, in der diese Änderung erfolgen soll, *nicht festgelegt*. An diesem Axiom kann die "Liberalität" des strukturalistischen Theorienkonzepts nochmals verdeutlicht werden. Obwohl das Axiom inhaltlich sehr vage ist, kann der Zusammenhang in strukturalistischer Deutung völlig exakt angegeben werden.

Damit ist der mathematische Strukturkern der Theorie vollständig festgelegt. Zu den bislang vorliegenden Strukturklassen M_p und M fehlt nun noch die Information, auf *was* diese Strukturen angewendet werden sollen. Da HT eine empirische Theorie

sein soll, muß sie einen bestimmten Realitätsausschnitt erfassen. Zu den bereits eingeführten Strukturen kommt deshalb nun als dritte Strukturklasse die Menge der intendierten Anwendungen I hinzu.

Wie oben bereits erwähnt, liefern Heider und andere (Experimental-)Psychologen viele Beispiele für die Anwendung der Theorie, die man als triadische Einstellungssysteme bezeichnen kann. Die ursprünglich von Heider genannte Menge I_0 wurde im historischen Verlauf allerdings kaum vergrößert, vielmehr ergaben sich Hinweise, daß nicht alle Einstellungssysteme intendierte Anwendungen sind.

Berger et al (1962: 9-36) beschreiben folgende Situation (vgl. auch Opp 1984: 32): in bestimmten Gemeinschaften existiert eine Norm, daß Männer Alkohol trinken sollen, während Frauen diesen meiden sollten. Angenommen, ein Mann liebt ein Frau und er mag Alkohol, während seine Frau Alkohol ablehnt. Diese Triade ist unbalanciert und führt nach der Theorie zu Druck auf Veränderung und Gleichgewicht. Tatsächlich befindet sich diese Situation jedoch vollständig in Übereinstimmung mit existierenden Normen, und es besteht deshalb mit hoher Wahrscheinlichkeit kein Veränderungsdruck. An diesem Beispiel zeigen sich noch einmal konkret die unterschiedlichen Auffassungen von Kritischen Rationalisten und Strukturalisten. Kritische Rationalisten wie Opp interpretieren dies als Falsifikation der Heider-Theorie (Opp 1984: 32) bzw. als Aufforderung, den Theoriekern zu modifizieren. In Anbetracht vieler positiver experimenteller Befunde und erfolgreicher Anwendungen der Theorie würde dies jedoch eine völlig überzogene Reaktion darstellen, die jeglicher Intuition widerspricht, und die kein Naturwissenschaftler in einer analogen Situation machen würde.

Aus strukturalistischer Perspektive ist der Theoriekern infolge der Offenheit von I immun gegen Widerlegung. Die berichtete Anomalie ist ein Hinweis darauf, daß stark normbestimmte Situationen nicht zum Anwendungsbereich der Heider-Theorie gehören und ausgeschlossen werden sollten. Dies ist auch die Interpretation von Berger et al (1962: 13): "These exceptions are not so much a gap in Heider's theory as an indication of the way in which its scope is to be defined and limited".[55] Heider selbst hat schon Balancesysteme aus dem Anwendungsbereich eliminiert, in denen Eifersucht, Neid und Wettbewerb eine Rolle spielen (Heider 1946: 110-111; 1977: 233). Eventuell müssen bei den intendierten Systemen noch weitere Einschränkungen gemacht werden, die mit einem vorsichtigen Gebrauch der Regel der Autodetermination ermittelt werden können. "Das sollte nicht ad hoc, sondern nach sorgfältigem Vergleich mit den bislang erfolgreichen Anwendungen geschehen, so daß ein systematischer Unterschied zwischen den erfolgreichen und den erfolglosen Anwendungen angegeben werden kann" (Stephan 1990: 84).

Definition 7 ist ein *Versuch*, die Menge der intendierten Anwendungen von HT allgemein zu charakterisieren.

[55] Vgl. hingegen die Reaktion von Opp (1984: 32-33), der die Bereitschaft, Anomalien zu akzeptieren bzw. aus dem Anwendungsbereich auszuschließen, kritisiert.

Definition 7
x ist eine intendierte Anwendung von HT ($x \in I(HT)$) gdw es O, T, <, P und N gibt, so daß gilt:
(1) $x = \langle O,T,<,P,N\rangle$
(2) $x \in M_p(HT)$
(3) O, P und N sind kognitiv repräsentierte Mengen in einem menschlichen Individuum
(4) Zwischen je zwei $o_1,o_2 \in O$ bestehen (kognitiv repräsentierte) Relationen, die als positiv oder negativ klassifiziert werden können (Einstellungen)
(5) x referiert nicht auf Situationen, die bestimmt sind von Normen, Eifersucht, Neid, Wettbewerb ...

Intendierte Anwendungen der Heider-Theorie sind also alle Personen, deren kognitive Systeme sich in der Begrifflichkeit der Theorie beschreiben lassen und insbesondere die Axiome (3), (4) und (5) von Definition 7 erfüllen. Axiom (5) sollte - unter Anwendung der Autodeterminationsregel - näher charakterisiert werden. Wichtig ist, daß die Menge der intendierten Anwendungen hier pragmatisch festgelegt ist und nicht etwa "formal definiert" wird. Es sei anhand dieses konkreten Beispiels nochmals darauf hingewiesen, daß die Menge der intendierten Anwendungen aus prinzipiellen Gründen nicht formal festgelegt werden kann. Dies würde die Theorie zu einer mathematischen Theorie machen.

Die letzte Definition faßt schließlich alle Strukturelemente in einem Prädikat "... ist Heiders Balancetheorie" zusammen.

Definition 8
$HT = \langle M, M_p, I\rangle$ ist *Heiders Balancetheorie* gdw
(1) M_p ist die Menge der potentiellen Modelle (Def. 4)
(2) M ist die Menge aller Modelle (Def. 6)
(3) I ist die Menge der intendierten Anwendungen (Def. 7)

Die drei Strukturelemente charakterisieren Heiders Balancetheorie vollständig: M_p definiert die von der Theorie verwendete Begrifflichkeit, M enthält das Fundamentalgesetz der Balancetheorie, und I legt die intendierten Anwendungen fest. Alle Personen mit Einstellungen, auf welche die Modelldefinition 6 angewendet werden kann, sind damit Modelle der Heider-Theorie.

6.3 Wissensbasierte Modellierung

Die Balancetheorie in der Fassung von Heider ist ein fruchtbares Ausgangsmodell, aber infolge seiner Einfachheit und der beschränkten Menge intendierter Anwendungen theoretisch nur bedingt interessant. Insbesondere bietet die Theorie auf dem Computer wenig. Die Datenbasis ist so klein und überschaubar, daß die einfachen Folgerungen sofort sichtbar sind. Wir möchten das Computermodell aber hier

benutzen, um die beiden unterschiedlichen Modellierungsarten - Datengenerierungs-Modell und strukturalistisches Modell - an diesem einfachen Beispiel zu erläutern.

6.3.1 Datengenerierungs-Modell

Die Datenbasis des Heider-Modells könnte aus Kognitionen in Form von einfachen Subjekt-Verb-Objekt-Sätzen bestehen, wobei das Verb eine L- oder U-Relation ausdrücken müßte und Subjekt bzw. Objekt Personen und Nichtpersonen sind. Um linguistische Satzanalysen zu vermeiden, muß gewährleistet sein, daß Kognitionen wie

```
(D1)  John is married to Berta. He prefers Democrates while Berta
      dislikes Democrates
```

oder

```
(D2)  John und Berta, beide miteinander verheiratet, sind
      entschiedene Rauchgegner
```

als PROLOG-aufbereitete Subjekt-Prädikat-Objekt-Sätze eingelesen werden:

```
(D1') john 'is married to' berta. john prefers democrates. berta
      dislikes democrates.
(D2') john 'is married to' berta. john hates smoking. berta hates
      smoking.
```

Die Voraussetzungen an die Datenbasis müßten durch entsprechende Prüfprädikate sichergestellt sein.

Ein Klassifikationsprädikat könnte dann die Klassifikation der Verben in die beiden Grundrelationen P und N übernehmen. Hierzu kann ein einfaches Lexikon bereitgestellt werden, mit dem die wichtigsten Verben in einer Liste gesammelt und in "positiv" oder "negativ" klassifiziert werden.

```
lexikon(positiv,[likes, loves, positiv, 'is married to', ...]).
lexikon(negativ,[hates,dislikes,...]).
```

Ist das Verb nicht in der Liste enthalten, kann an den Benutzer bezüglich der Relationsbewertung eine Frage gestellt werden.
Den Relationen P und N entsprechen die beiden 2-stelligen PROLOG-Prädikate `positiv/2` und `negativ/2`, so daß aus (D1) die PROLOG-Fakten

```
positiv(john,berta).
positiv(john,democrates).
negativ(berta,democrates).
```

entstehen würden.

Damit kommen wir zu den interessanteren Prädikaten. Für die in Definition 3 festgelegten Triadenteilmengen G, U, I $\subseteq$ TR benutzen wir den Namen `triade/2` mit der Struktur an erster Argumentstelle und dem Triadenzustand G, U oder I an zweiter Stelle. Die Regeln nach Definition 3 könnten dann wie folgt repräsentiert werden:

```
triade((positiv(P,O), positiv(P,X), positiv(O,X)), gleichgewicht)
   :-
   positiv(P,O),
   positiv(P,X),
   positiv(O,X).

triade((positiv(P,O), negativ(P,X), negativ(O,X)), gleichgewicht)
   :-
   positiv(P,O),
   negativ(P,X),
   negativ(O,X).

triade((positiv(P,O), negativ(P,X), positiv(O,X)), ungleichgewicht)
   :-
   positiv(P,O),
   negativ(P,X),
   positiv(O,X).
...
```

Die erste Regel kann gelesen werden als: Die Struktur ... ist eine Triade im Gleichgewicht, wenn zwischen allen Objekten positive Relationen bestehen. Wir können nun an die Wissensbasis bereits Fragen nach den Eigenschaften von Triaden stellen. Zur Verdeutlichung sind PROLOG-Fragen im weiteren Verlauf der Arbeit fett gedruckt. Das Ergebnis einer Anfrage an die Datenbasis (D1) mit

```
?- triade(X,Y).
```

wäre dann

```
X = (positiv(john,berta), positiv(john,democrates),
    negativ(berta,democrates))
Y = ungleichgewicht
```

Ein Beweisversuch zur Herleitung einer gleichgewichtigen Triade in (D1) schlägt fehl:

```
?- triade(X,gleichgewicht).
no
```

Auf der Grundlage dieser Regeln lassen sich weitere Regeln bezüglich Stabilität und Instabilität von Triaden einführen.

```
triade((X,instabil) :-
   triade(X,ungleichgewicht).

triade(X,stabil) :-
   triade(X,gleichgewicht).
```

Die erste Regel besagt, daß eine Triade X instabil ist, wenn sie ungleichgewichtig ist, und die zweite besagt, daß eine Triade X stabil ist, wenn sie im Gleichgewicht ist.

Schließlich läßt sich ein Prädikat `tendenz/2` definieren, das die Möglichkeiten der Entwicklung unbalancierter in balancierte Strukturen beinhaltet. `tendenz(X,Y)` ist zu lesen als: Die Struktur X hat eine Tendenz zu Y.

```
tendenz(X,X) :-
   triade(X,stabil).
```

```
tendenz(X,Y) :-
   triade(X,instabil),
   transformation(X,Y).
```

Das erste `tendenz`-Prädikat besagt, daß sich die Struktur tendenziell nicht verändert, wenn X eine stabile Triade ist. Das zweite Tendenz-Prädikat bedeutet, daß sich X tendenziell nach Y verändert, wenn X instabil ist und eine Transformation von X nach Y möglich ist.

Das Transformations-Prädikat überführt unbalancierte Strukturen in balancierte, wobei hier als Möglichkeit nur die Änderung der *Relationen* in Betracht gezogen wurde. Für jede Struktur gibt es drei Möglichkeiten, so daß mit `transformation(X,Y)` jedem X drei Y-Strukturen zugeordnet sind. Die Transformations-Prädikate für Triade (e) von Abb. 6.2 lauten dann beispielsweise:

```
transformation((positiv(P,O),negativ(P,X),positiv(O,X)),
               (positiv(P,O),positiv(P,X),positiv(O,X))).
transformation((positiv(P,O),negativ(P,X),positiv(O,X)),
               (negativ(P,O),negativ(P,X),positiv(O,X))).
transformation((positiv(P,O),negativ(P,X),positiv(O,X)),
               (positiv(P,O),negativ(P,X),negativ(O,X))).
```

Mit (D2) als Datenbasis wird die Anfrage

```
?- tendenz(X,Y).
```

beantwortet mit

```
X = (positiv(john,berta), negativ(john,rauchen),
    negativ(berta,rauchen))
Y = (positiv(john,berta), negativ(john,rauchen),
    negativ(berta,rauchen)).
```

Dies bedeutet, die Relationen ändern sich nicht über die Zeit (da die Datenbasis schon balanciert ist). Mit der ungleichgewichtigen Triade (D1) als Datenbasis ergeben sich drei Alternativen für eine Relationenänderung:

```
X = (positiv(john,berta), positiv(john,democrates),
     negativ(berta,democrates))
Y = (positiv(john,berta), negativ(john,democrates),
     negativ(berta,democrates))
;
Y = (positiv(john,berta), positiv(john,democrates),
     positiv(berta,democrates))
;
Y = (negativ(john,berta), positiv(john,democrates),
     negativ(berta,democrates))
;
no
```

Die eben vorgestellte Implementierung ist ein Datengenerierungs-Modell, weil das Modell neue Daten erzeugt, die z.B. mit realen Systemen verglichen werden können. In der nächsten Version erzeugt das Modell keine neuen, empirisch direkt interpretierbaren Daten.

6.3.2 Strukturalistische PROLOG-Prädikate

In der strukturalistisch ausgerichteten Implementierung soll das Programm die "deklarative Rolle" der Axiome übernehmen und Gültigkeitsüberprüfungen vornehmen. In diesem Fall repräsentieren die PROLOG-Regeln also programmiersprachlich die strukturalistischen Prädikate. Der Benutzer gibt die Beschreibung einer Struktur als Datenbasis vor und kann mit den PROLOG-Regeln feststellen, ob die Prädikate in der vorliegenden Datenbasis gelten oder nicht gelten. Die Inputdaten werden hier also von einer Struktur gebildet, das Programm prüft, ob sie die Axiome erfüllt und liefert als Outputdaten Wahrheitswerte. Im Gegensatz zur ersten Version werden bei dieser Variante keine neuen Modelldaten generiert. Man kann die so erzeugten Outputs wie "x ist ein potentielles Modell" etc. allenfalls als Meta-Daten interpretieren.

Wenn wir uns bei der Implementierung an die eingeführten strukturalistischen Prädikate halten, müssen wir die Datenbasis etwas ändern, da potentielles Modell und Modell explizit auf die Zeit Bezug nehmen. Im potentiellen Modell und im Modell betrachten wir eine endliche Menge von Zeitpunkten und für jeden dieser Zeitpunkte die Relationen zwischen je zwei Elementen der Objektmenge.

Unsere vom Benutzer einzugebenden Grundprädikate bilden zunächst das Prädikat `zeit/1`, welches den betrachteten Zeitraum in einer Liste festlegt und das Prädikat `objekte/1`, in dem die Menge der betrachteten Objekte festgelegt wird.

```
zeit([1,4,5,8]).
objekte([john,democrates,berta]).
```

Der Benutzer muß als weitere Grundbegriffe die P- und N-Relationen definieren und gibt diese in Form einfacher Subjekt-Verb-Objekt-Sätze ein, zusammen mit dem Zeitpunkt, zu dem sie gelten. Das Einlesen funktioniert im Prinzip nach dem oben genannten Verfahren, in dem ein Klassifikator für die Einordnung des Verbs in die Mengen P oder N sorgt. Die Relationen werden dann zusammen mit dem Zeitpunkt, zu dem sie bestehen, abgespeichert, so daß wir z.B. folgende Datenbasis erhalten:

```
positiv(1,john,berta).
positiv(1,john,democrates).
negativ(1,berta,democrates).
positiv(4,john,berta).
positiv(4,john,democrates).
negativ(4,berta,democrates).
positiv(5,john,berta).
positiv(5,john,democrates).
positiv(5,berta,democrates).
positiv(8,john,berta).
positiv(8,john,democrates).
positiv(8,berta,democrates).
```

Die Einleseprozeduren können dem Anhang entnommen werden. Bei der Eingabe der Relationen können natürlich einerseits Zeitpunkte nicht in der Zeitliste enthalten sein und andererseits Objekte nicht in der Objektliste. Dies zu überprüfen ist u.a. Aufgabe der strukturalistischen Prädikate. Das Prädikat für den Heider-Graphen wird z.B.

schon dann scheitern, wenn die Kardinalität der Objektliste $\neq 3$ ist. Faktenbasen der eben angegebenen Art sind also nun Prüfkandidaten für die strukturalistischen Prädikate.

Wir versuchen uns im folgenden möglichst genau an die vorgegebenen strukturalistischen Prädikate zu halten unter Verzicht auf Effizienz und knappen Code in PROLOG. Das PROLOG-Prädikat `heider_graph/2` definiert den Heider-Graphen gemäß Definition 1, wobei wir gleich den Zeitindex hinzugenommen haben. `heider_graph(T,X)` ist also zu lesen als: (das Tupel) `X` ist zu `T` ein Heider-Graph. `T` ist hierbei wegen PROLOG-Konvention eine Variable, die einem $t \in T$ der obigen Definitionen entspricht. Der Menge T entspricht in den PROLOG-Prädikaten die Variable `T_liste` (Text hinter `%` ist Kommentar).

```
heider_graph(T,X) :-                    % X ist Heider-Graph zu Zeit T, wenn
   X = [O,P,N],                         % X = [O,P,N] und
   objekte(O),                          % O ist die Menge der Objekte und
   not O=[],                            % O ist nicht leer und
   O = [O1,O2,O3],                      % O hat die Kardinalität 3 und
   atom(O1),atom(O2),atom(O3),          % die Elemente von O sind Atome und
   zeit(T_liste),                       % T_liste ist die betracht. Zeitmenge und
   member(T,T_liste),                        % T ist Element von T_liste und
   findall([X1,X2],positiv(T,X1,X2),P), % P i.d.Menge d.positiv. Relat. und
   findall([Y1,Y2],negativ(T,Y1,Y2),N), % N i.d.Menge d.negativ. Relat. und
   kreuzprodukt(O,O,Kreuz)              % Kreuz i.d.Cartes. Produkt O x O und
   teilmenge(P,Kreuz),                  % P ⊆ Kreuz und
   teilmenge(N,Kreuz),                  % N ⊆ Kreuz und
   durchschnitt(P,N,[]),                % P ∩ N = ∅ und
   vereinigung(P,N,Rel),                % P ∪ N = Rel und
   irreflexiv(O,Rel),                   % Rel ist irreflexiv und
   vollstaendig(Kreuz,Rel),!.           % zwischen allen Paarelem. existiert Rel.
```

Das PROLOG-Prädikat `heider_graph/2` entspricht ziemlich genau Definition 1. `objekte(O)` instantiiert die Objektmenge. Die nachfolgenden fünf Clausen (bis `atom(O3)`) überprüfen die Objektmenge `O`. Die Clause `not O = []` ist eigentlich überflüssig, da sie von der nächsten Clause impliziert wird und wurde nur zur Verdeutlichung eingefügt. Die PROLOG-Built-in-Prädikate `atom(O1),...,atom(O3)` stellen sicher, daß die Elemente von `O` weder Listen noch Zahlen noch Variable sind (d.h. `atom(X)` scheitert, wenn `X=[a,b]`, `X=3` oder `X=Y`), sondern Atome wie z.B. `a`, `hans`, `uni_muenchen`. Die Prädikate `zeit(T_liste)` und `member(T,T_liste)` sorgen dafür, daß der Heider-Graph nur für die vorgegebenen Zeitpunkte betrachtet wird. Das Built-in-Prädikat `findall/3` sammelt alle Paare, zwischen denen eine positive bzw. negative Relation besteht in eine Liste `P` bzw. `N`, so daß `P` und `N` genau die in Axiom (3) $P \subseteq O \times O$ und (4) $N \subseteq O \times O$ festgelegten Teilmengen enthalten. Die vier Hilfsprädikate `kreuzprodukt/3`, `teilmenge/2`, `durchschnitt/3` und `vereinigung/3` definieren die entsprechenden mengensprachlichen Operationen. Deren Arbeitsweise kann dem Anhang entnommen werden. Die Axiome (6) und (7) von Definition 1 beziehen sich auf die Vereinigung von P und N, welche zunächst mit dem Vereinigungsprädikat durchgeführt wird. Axiom (6) entspricht dem Hilfsprädikat `irreflexiv/2`, das erfolgreich ist, wenn die Relation $Rel = P \cup N$ irreflexiv ist:

```
irreflexiv([],_).
irreflexiv([X|R],Rel) :-
   not member([X,X],Rel),
   irreflexiv(R,Rel).
```

Die letzte Clause `vollstaendig(Kreuz,Rel)` verifiziert mit Hilfe der Kreuzproduktliste schließlich, daß zwischen allen Paarelementen `[X,Y]` - mit Ausnahme von X=Y - eine P- oder N-Relation besteht (vgl. Anhang). Da das `heider_graph`-Prädikat für jeden Heider-Graphen nur einmal erfüllbar sein soll, ist ein Cut zur Verhinderung von Backtracking eingefügt.

`heider_graph/2` kann nun benutzt werden, um für eine beliebige Datenbasis für einen beliebigen Zeitpunkt festzustellen, ob diese ein Heider-Graph ist. Hierzu versucht der PROLOG-Interpreter alle Teilziele des Prädikats `heider_graph/2` zu beweisen. Bei instantiiertem Zeitpunkt `T` und uninstantiiertem zweiten Argument liefert das Prädikat im negativen Fall `no` - d.h. das Datenbeispiel ist kein Heider-Graph - im positiven Fall gibt es das Tupel `[O,P,N]` zu `T ∈ T_liste` aus.

Beispielsweise liefert

```
?- heider_graph(2,X).
no
```

weil der Zeitpunkt 2 nicht in der Zeitliste enthalten ist und die Clause `member(T,T_liste)` damit scheitert.

Für

```
?- heider_graph(4,X).
```

gelingt jedoch der Beweis mit folgender Ausgabe:

```
X=[[john,democrates,berta],
  [[john,berta],[john,democrates]],[[berta,democrates]]].
```

Die erste Teilliste ist hierbei die Objektmenge, und die zweite Teilliste ist die Liste der Relationen mit den positiven Relationen als erster und den negativen Relationen als zweiter Teilliste.

Werden die drei Fakten

```
positiv(9,john,berta).
positiv(9,a,b).
positiv(9,berta,john).
```

hinzugefügt, so scheitert `heider_graph(9,X)` weil `[a,b]` nicht Element des Kreuzprodukts aus O × O ist (genauer: die Clause `teilmenge(P,Kreuz)` scheitert).

Auf einen grundlegenden Unterschied zwischen den mengensprachlichen Definitionen und den PROLOG-Regeln müssen wir an dieser Stelle noch hinweisen. Während in der mengensprachlichen Definition aufgrund der logischen Äquivalenz ("gdw") sowohl der Schluß von links nach rechts als auch umgekehrt möglich ist, können wir in PROLOG nur von rechts nach links ableiten. Am Beispiel: PROLOG schließt aus der Existenz des Tupels X mit der Objektmenge O usw. auf einen Heider-Graphen; es ist aber in PROLOG nicht wie in der mengensprachlichen Definition möglich, vom Heider-Graphen auf die Existenz von X, O etc. zu schließen. Logische Äquivalenzen sind in PROLOG zumindest nicht innerhalb *einer* Regel darstellbar. Dieser Unterschied betrifft grundsätzlich alle folgenden Definitionen.

Die nächsten Prädikate entsprechen Definition 2:

```
r(T,R) :-                        % R i.d.Menge aller Relationen zum Zeitp. T wenn
   heider_graph(T,[_,P,N]),      % [_,P,N] Heider-Graph zu T ist und
   vereinigung(P,N,R).           % R = P ∪ N.

np(T,X) :-                       % X ist Nicht-Person zum Zeitpunkt T wenn
   heider_graph(T,[O,P,N]),      % [O,P,N] Heider-Graph zu T ist und
   member(X,O),                  % X Element von O ist und
   not positiv(T,X,_),           % X nicht Anfangspunkt d.positiv. Rel.zu T ist und
   not negativ(T,X,_).           % X nicht Anfangspunkt d.negativen Rel. zu T ist.

pe(T,X) :-                         % X ist eine Person zum Zeitpunkt T wenn
   heider_graph(T,[O,_,_]),        % [O,_,_] Heider-Graph zu T ist und
   np(T,Y),                        % Y Nicht-Person zu T ist und
   differenz(O,[Y],X_liste),       % X_liste Diff.menge zw. O und [Y] ist und
   member(X,X_liste).              % X Element aus X_liste ist.
```

Beispielsweise sind damit folgende Anfragen möglich:

Welche Personen sind zum Zeitpunkt 1 Element des Heider-Graphen?

```
?- pe(1,X).
X = john;
X = berta;
no
```

Alle Nichtpersonen zum Zeitpunkt 1:

```
?- np(1,X).
X = democrates;
no
```

Zu welchem Zeitpunkt sind John und a Personen?

```
?- pe(X,john).
X = 1;
X = 4
?- pe(X,a).
no
```

Definition 3 legt balancierte, unbalancierte und indefinite Triaden fest, die in acht Regeln definiert werden können. Wir geben hier nur zwei wieder.

```
triade(T,O,gleichgewicht) :-      % O ist zu T Triade im Gleichgewicht wenn
   heider_graph(T,[O,P,N]),       % [O,P,N] Heider-Graph zu T ist und
   member(A,O),                   % A Element von O ist und
   pe(T,A),                       % A zu T eine Person ist und
   member(B,O),                   % B Element von O ist und
   not B = A,                     % A und B verschieden sind und
   pe(T,B),                       % B zu T eine Person ist und
   member(C,O),                   % C Element von O ist und
   np(T,C),                       % C zu T eine Nichtperson ist und
   member([A,B],P),               % [A,B] Element der P-Relation ist und
   member([A,C],P),               % [A,C] Element der P-Relation ist und
   member([B,C],P),!.             % [B,C] Element der P-Relation ist.
```

```
triade(T,O,ungleichgewicht) :-
   heider_graph(T,[O,P,N]),
   member(A,O),
   pe(T,A),
   member(B,O),
   not B = A,
   pe(T,B),
   member(C,O),
   np(T,C),
   member([A,B],N),
   member([A,C],P),
   member([B,C],P),!.
```

Das erste Teilziel prüft jeweils, ob zu `T` ein Heider-Graph vorliegt. Die nächsten sieben Prädikate sorgen dafür, daß die richtigen Elemente aus der Objektmenge `O` instantiiert werden: `A` und `B` müssen Personen sein, die verschieden sind und `C` muß eine Nicht-Person sein. Erst mit den letzten drei `member`-Prädikaten wird die Zuordnung zu Gleichgewichts-, Ungleichgewichts- und Indifferenzzuständen getroffen. Da jede Triade genau einer der acht Regeln zugeordnet werden kann, braucht im Erfolgsfall nicht noch eine andere Regel geprüft zu werden, so daß am Ende jeder Regel ein Cut zur Verhinderung von Backtracking eingefügt wird.
Die Anfrage nach dem Zustand der Triade zum Zeitpunkt 4 ist beispielweise:

```
?- triade(4,X,Zustand).
X        = [john,democrates,berta]
Zustand = ungleichgewicht
```

Bei der Regel für das potentielle Modell können wir uns wieder ziemlich eng an Definition 4 anlehnen. Allerdings müssen wir hier für einzelne Axiome wieder Hilfsprädikate einführen.

```
pot_modell(X) :-                          % X ist ein potentielles Modell wenn
   X=[O,T,'<',Pt,Nt],                     % X das Tupel [O,T,'<',Pt,Nt] ist und
   objekte(O),                            % O die Objektmenge ist und
   O=[O1,O2,O3],                          % O dreielementig ist und
   atom(O1),atom(O2),atom(O3),            % die Elemente von O Atome sind und
   zeit(T_liste),                         % T_liste die Menge der Zeitpunkte ist und
   lineare_ordnung(T_liste,'<'),          % (T_liste,'<') e. lineare Ordnung ist und
   kreuzprodukt(O,O,Kreuz),               % Kreuz d. Kreuzprodukt von O und O ist und
   potenzmenge(Kreuz,Potmenge),           % Potmenge die Potenzmenge v.Kreuz ist und
   funktion(positiv,T_liste,Pt,Potmenge),     % Pt: T_liste → Potmenge und
   funktion(negativ,T_liste,Nt,Potmenge),     % Nt: T_liste → Potmenge und
   sind_alle_heider_graphen(T_liste,[O,Pt,Nt]). % [O,Pt,Nt] alle Heider-
                                                % Graph. zu T_liste sind.
```

Zunächst wird wieder die Objektmenge abgeprüft. Das Prädikat `lineare_ordnung/2` kontrolliert, ob die Liste der Zeitpunkte sortiert nach der Kleiner-Relation vorliegt. Es ist so allgemein geschrieben, daß jede beliebige Ordnungsrelation einer numerischen Liste geprüft werden kann:

```
lineare_ordnung([X,Y], Rel) :-
   Z =.. [Rel,X,Y],
   call(Z).
```

```
lineare_ordnung([X,Y|R],Rel) :-
   Z =..[Rel,X,Y],
   call(Z),
   lineare_ordnung([Y|R],Rel).
```

Die Relation - z.B. Kleiner, Größer, Kleiner/Gleich etc. - wird an zweiter Argumentstelle übergeben. Der Univ-Funktor =.. macht aus der mit der Relation und beiden Zahlen konstruierten Liste `[Rel,X,Y]` ein Prädikat: beispielweise wird so aus `[<,3,5]` das Prädikat `<(3,5)`. Bei einem Ziel, in dem der Funktor eine instantiierte Variable ist, muß das Ziel anschließend mit `call/1` aufgerufen werden. Das Prädikat `lineare_ordnung/2` ruft sich rekursiv so lange auf, bis die Ordnungsrelation entweder nicht erfüllt ist - in diesem Fall scheitert das Prädikat - oder die abgearbeitete Liste nur mehr aus zwei Elementen besteht, die der Ordnungsrelation genügen - in diesem Fall ist das Prädikat erfolgreich.

`kreuzprodukt/3` und `potenzmenge/2` sind wieder die entsprechenden mengensprachlichen Prädikate. Das nächste Prädikat `funktion/4` wird mit `positiv` oder `negativ` an erster Argumentstelle aufgerufen und berechnet für jeden Zeitpunkt die zugeordneten positiven und negativen Relationen.

```
funktion(Rel,[],[],_).
funktion(Rel,[T|T_rest],[(T,L)|R1], Potmenge) :-
   Z =.. [Rel,T,X1,X2],
   findall([X1,X2],Z,L),
   teilmenge(L,Potmenge),
   funktion(Rel,T_rest,R1,Potmenge).
```

Die Listen `Pt` und `Nt` enthalten nach Abarbeitung der `funktion`-Prädikate Mengen von Paaren `(T,L)`, wobei jedem Zeitpunkt `T` (erstes Element) die Menge der zu `T` bestehenden Relationen `L` zugeordnet ist (zweites Element).

```
Pt: [(1,[[john,berta], [john,democrates]]), (4,[[john,berta], ...]
Nt: [(1,[[berta,democrates]]), (4,[[berta,democrates]],...]
```

Schließlich fordert das zentrale fünfte Axiom von Definition 5, daß für alle betrachteten Zeitpunkte ein Heider-Graph vorliegen muß. Auch hier müssen wir ein Hilfsprädikat definieren, das die Zeitpunkte rekursiv abarbeitet und für jeden Zeitpunkt verifiziert, daß ein Heider-Graph gegeben ist:

```
sind_alle_heider_graphen([],[_,[],[]]).
sind_alle_heider_graphen([T|T_rest],[O,[(T,P)|P_rest],
                        [(T,N)|N_rest]]) :-
   heider_graph(T,[O,P,N]),!,
   sind_alle_heider_graphen(T_rest,[O,P_rest,N_rest]).
```

`sind_alle_heider_graphen/2` arbeitet die Zeitliste rekursiv zusammen mit `Pt` und `Nt` ab und verifiziert für jeden Zeitpunkt, daß ein Heider-Graph vorliegt. Das Prädikat ist nur erfolgreich, wenn für *alle* Zeitpunkte der vorgegebenen Menge Heider-Graphen existieren.

In unserem Beispiel liefert

```
?- pot_modell(X).
```

die folgende Struktur und damit den Beweis, daß die Datenbasis ein potentielles Heider-Modell ist:

```
X  = [
       [john,democrates,berta],                          % Objektliste
       [1,4,5,8,9],                                      % Zeitpunkte
       '<',                                              % Ordnungsrelation
       [
         (1,[[john,berta],[john,democrates]]),           % P-Relation zu T=1
         (4,[[john,berta],[john,democrates]]),           %            zu T=4
         (5,[[john,berta],[john,democrates],[berta,democrates]]),
         (8,[[john,berta],[john,democrates],[berta,democrates]])
       ],
       [
         (1,[[berta,democrates]]),                       % N-Relation zu T=1
         (4,[[berta,democrates]]),                       %            zu T=4
         (5,[]),                                         %            zu T=5
         (8,[])                                          %            zu T=8
       ]
     ]

yes
```

Schließlich läßt sich in Anlehnung an Definition 6 das PROLOG-Modell-Prädikat einführen, das in PROLOG etwas umgeschrieben werden muß.

```
modell(X) :-
   X=[O,T_liste,_,_,_],
   pot_modell(X),!,
   axiom_3_test(T_liste, T_liste,O).


axiom_3_test([],_,_).
axiom_3_test([T1|T_rest],T_liste,O) :-
   triade(T1,O,ungleichgewicht),
   member(T2,T_liste),
   T1 < T2,
   triade(T2,O,gleichgewicht),!,
   axiom_3_test(T_rest,T_liste,O).
axiom_3_test([T1|T_rest],T_liste,O) :-
   triade(T1,O,gleichgewicht),
   member(T2,T_liste),
   T1 >= T2,
   triade(T2,O,ungleichgewicht),!,
   axiom_3_test(T_rest,T_liste,O).
```

Zunächst verifiziert `modell(X)`, daß `X` ein potentielles Modell ist. Für den Test von Axiom (3) wird ein eigenes Prädikat eingeführt, das alle Elemente der Zeitliste abarbeitet (wegen Allquantor über T). Das Prädikat ist erfüllt, wenn es für jeden Zeitpunkt `T1`, zu dem eine ungleichgewichtige Triade existiert, einen Zeitpunkt `T2` > `T1` gibt, zu dem die Triade gleichgewichtig ist. Man beachte, daß `T2` automatisch durch PROLOG-Backtracking gesucht und bestimmt wird. Ist die Triade zu `T1` im Gleichgewicht, so ist die Bedingung von Axiom 3 nicht erfüllt. Da wir aber die Zeitliste vollständig abprüfen, müssen wir auch diesen Fall noch berücksichtigen, der der dritten Clause von `axiom_3_test` entspricht. Sie besagt folgendes: ist eine Triade zu `T1` im

Gleichgewicht, dann gab es einen Zeitpunkt `T2` $\leq$ `T1`, zu dem sie ungleichgewichtig war.

In unserem Beispiel liefert das Modell-Prädikat den Beweis, daß mit der Datenbasis ein Heider-Modell vorliegt, und es wird - bei Aufruf mit Variable - das Tupel wie oben ausgegeben. Scheitert das Modell-Prädikat nach Beweis von `pot_modell(X)`, so liegt zwar ein potentielles Heider-Modell vor, aber das grundlegende inhaltliche Axiom gilt nicht in der Datenbasis. Wären alle Relationen in der kleinen Datenbasis über alle Zeitpunkte z.B. immer positiv, ist `pot_modell(X)` erfüllt, aber `modell(X)` scheitert.

Der Theoriekern der strukturalistisch interpretierten Heider-Theorie liegt damit als PROLOG-Programm vor. Die Prädikate könnten noch etwas ausgebaut und informativer aufbereitet werden, die Ausgaben leserlicher gestaltet und eine Erklärungskomponente hinzugefügt werden. Die Erklärungskomponente sollte insbesondere Rechtfertigungen darüber liefern, warum ein Prädikat zutrifft bzw. nicht zutrifft, also z.B. warum eine Faktenbasis ein potentielles Modell, aber kein Modell ist. In Anbetracht der Einfachheit des Modells erscheint dies aber nicht sehr lohnenswert, und wir wollen uns nun etwas komplexeren Modellen zuwenden.

7. Symbolic Psycho-Logic: Die Balancetheorie von Abelson/Rosenberg

Eine bekannte Generalisierung der Theorie von Heider ist das Balancemodell von Abelson/Rosenberg (1958). Es gehört zusammen mit dem Modell von Newcomb (1953) - auf das wir nicht eingehen - zu den klassischen, kognitiv orientierten Gleichgewichtstheorien und wird in vielen Lehrbüchern beschrieben (z.B. Witte 1989, Irle 1975).

7.1 Informelle Darstellung und logische Struktur

Abelson/Rosenberg (1958) erweitern das Grundmodell von Heider in dreierlei Hinsicht. Erstens werden mehr als drei Objekte zugelassen, zweitens gibt es nun drei Grundrelationen und eine abgeleitete Relation, und drittens macht das Modell spezifische Aussagen darüber, wie unbancierte in balancierte Zustände überführt werden.

Die Objekte und Relationen bilden bei Abelson/Rosenberg einen Einstellungsraum, welchen die Autoren als "conceptual arena" bezeichnen. Im Gegensatz zu den triadischen Einstellungsräumen des Grundmodells können diese nun einen *mehr oder weniger hohen Balancegrad* aufweisen. Das zentrale inhaltliche Axiom des Modells ist analog wie im Heider-Modell die Behauptung, daß unbalancierte Strukturen in balancierte geändert werden. Zusätzlich wird nun jedoch spezifiziert, *wie* diese Änderungen erfolgen.

Die Erweiterung des Grundmodells auf mehr als drei Objekte macht eine Implementierung auf dem Computer interessanter, da die Schlußfolgerungen nicht mehr so einfach und überschaubar sind wie bei Heider. Durch die Spezifizierung der Transformation unbalancierter in balancierte Strukturen behebt die Theorie zudem die bei Heider vorhandene Vagheit. Beibehalten wird die individuell-kognitive Sichtweise, nach der Relationen zwischen *kognitiven* Konfigurationen eines Individuums bestehen.

7.1.1 Grundbegriffe, Datenerhebung, Strukturmatrix

Betrachtet werden Repräsentationen von konkreten oder abstrakten Entitäten, die als kognitive *Elemente* oder *Objekte* bezeichnet werden. Wie bei Heider wird angenommen, daß Individuen sich auf diese Elemente mit verbalen Bezeichnern beziehen können. Im Einstellungsraum "Kirche" sind solche Elemente z.B. "Papst", "Pille", "Zölibat".

In dem Modell gibt es nun vier *Relationen*, die zwischen je zwei Objekten bestehen können: wie bei Heider eine positive (P) und negative (N) Grundrelation, sowie zusätzlich eine neutrale (O) und die mit P und N definierte ambivalente Relation (A). *Kognitive Einheiten* oder kurz: *Kognitionen* sind aufgebaut aus je zwei Objekten und einer Relation zwischen den Objekten. aNb wäre also eine Kognition, bei der

zwischen den Elementen a und b eine negative Relation besteht.

Wir führen gleich die strukturalistischen Prädikate zusammen mit der informellen Darstellung der Theorie ein. Die Rekonstruktion des Modells erfolgt dabei analog zum Vorgehen bei Heider. Zunächst werden die verwendeten Grundbegriffe und Relationen unabhängig von der Zeit in einer eigenen Definition "... ist ein Abelson-Rosenberg-Graph" charakterisiert, und anschließend wird dieses Prädikat als Definiens im potentiellen Modell zusammen mit der Zeit und der auf ihr definierten Ordnungsrelation verwendet.

Definition 1

x ist ein *Abelson-Rosenberg-Graph* ($x \in ARG$) gdw es X, P, N und O gibt, so daß gilt:

(1) $x = \langle X,P,N,O \rangle$
(2) X ist eine nicht-leere Menge (von kognitiven Elementen)
(3) $P \subseteq X \times X$
(4) $N \subseteq X \times X$
(5) $O \subseteq X \times X$
(6) P, N, O sind paarweise disjunkt
(7) Für alle $x \in X$: xPx
(8) Für alle $x,y \in X$: wenn xPy dann yPx
(9) Für alle $x,y \in X$: wenn xNy dann yNx
(10) Für alle $x,y \in X$: wenn xOy dann yOx

Die Definition des Abelson-Rosenberg-Graphen besagt, daß es eine nicht-leere Menge X von kognitiven Elementen und paarweise disjunkte Mengen gibt, nämlich die auf X definierten Relationen P, N und O. Die Relation P ist reflexiv, die Relationen P, N und O sind symmetrisch. Inhaltlich ausgedrückt bedeutet dies, daß jedes Element in einer positiven Beziehung zu sich selbst steht und, falls a und b in einer positiven (negativen, neutralen) Relation stehen, auch b und a in positiver (negativer, neutraler) Relation zueinander stehen.

Bevor wir auf die Balancefrage zu sprechen kommen, muß kurz die Methode der Datenerhebung vorgestellt werden.

Ziel der Datenerhebung ist die Gewinnung einer Strukturmatrix, deren Zellen aus Relationen $p^* \in P$, $n^* \in N$ oder $o^* \in O$ bestehen, welche zwischen je zwei Elementen $a_i,b_j \in X$ bestehen. Tab. 7.1 zeigt das Schema einer solchen Strukturmatrix.

Tab 7.1: Strukturmatrix mit den Elementen p*, n*, o*

	b_1	b_2	b_3 ...
a_1	p*	n*	o*
a_2	n*	p*	n*
a_3	o*	n*	p*
...			

Zum Aufbau der Strukturmatrix wird dem Probanden zunächst ein bestimmter Einstellungsraum vorgegeben, für den er die Objekte $a_i \in X$ (Wörter, Phrasen) auflistet.

Beispielsweise könnten zum Einstellungsraum "Kirche" folgende Elemente genannt werden: "Papst", "Pornographie", "Drewermann", "Zölibat". Abelson/Rosenberg verwenden als Thema "Having an honor system at Yale" mit folgenden Objekten: "The honest student", "Reporting cheaters", "Feeling trusted", "Cheating by a few", "an honor system". Wir benutzen dieses Beispiel, um später daran unser Modell zu testen.

Nach der Definition der Objektmenge werden alle Relationen zwischen den Objektpaaren erhoben. Hierzu wird dem Probanden eine Teilmenge des Cartesischen Produkts $X \cup \{ego\} \times X \cup \{ego\}$ in Form eines Lückentests vorgelegt. Der Proband soll die Lücken mit entsprechenden Relationen auffüllen. Beispielsweise könnten die Paare

```
Papst ..... Pornographie
Papst ..... Zölibat
```

wie folgt gefüllt werden:

```
Papst verabscheut Pornographie
Papst befürwortet Zölibat.
```

Im Originalbeispiel wurden u.a. folgende Relationen eingetragen:

```
The honest student feels very reluctant about reporting cheaters.
The honest student posseses the feeling of being trusted.
The honor system is fine for the honest student.
The honor system promotes the feeling of being trusted.
Cheating by a few is cut down by reporting cheaters.
The honor system is harmed by cheating by a few.
The honor system involves reporting cheaters.
```

Da das Kreuzprodukt sowohl $\langle x,x \rangle$ als auch $\langle x,y \rangle$ und $\langle y,x \rangle$ als Elemente enthält, P aber reflexiv ist und P, N, O symmetrisch sind, brauchen zum Ausfüllen der Strukturmatrix nicht alle Relationen zwischen den Elementpaaren des Kreuzprodukts empirisch erhoben werden. Vielmehr sind die Diagonalelemente der Matrix im vorhinein ebenso bekannt wie die zu xRy ($R := P \cup N \cup O$) symmetrische Relation yRx. Nach der Rohdatenerhebung werden die Verbalphrasen der Sätze klassifiziert in positive, negative und neutrale Relationen. Alternativ könnten natürlich die Versuchspersonen gleich angewiesen werden, lediglich die drei Modellrelationen zu benutzen.

Das Ergebnis der Datenerhebung ist die Strukturmatrix, die im Originalbeispiel wie folgt aussieht:

Tab. 7.2: Strukturmatrix des Originalbeispiels von Abelson/Rosenberg

	Ego	The honest stud.	Report. cheaters	Feel. trusted	Cheat. by a few	Honor System
Ego	p*	p*	n*	p*	n*	o*
The honest student	p*	p*	n*	p*	o*	p*
Reporting cheaters	n*	n*	p*	o*	n*	p*
Feeling trusted	p*	p*	o*	p*	o*	p*
Cheating by a few	n*	o*	n*	o*	p*	n*
An honor system	o*	p*	p*	p*	n*	p*

Man beachte, daß infolge der Symmetrie von P, N und O die Matrix symmetrisch ist und daß wegen der Reflexivität von P alle Diagonalelemente mit p* gefüllt sind.

Definition 2 definiert die Strukturmatrix formal.

Definition 2
Für $x \in ARG$ wird die Strukturmatrix S wie folgt definiert:
(1) $S: X \times X \rightarrow \{p^*,n^*,o^*\}$

(2) Für alle $x,y \in X$: $S(x,y)= \begin{cases} p^*, \text{ falls } xPy \\ n^*, \text{ falls } xNy \\ o^*, \text{ falls } xOy \end{cases}$

Die Matrixeigenschaften Symmetrie und Reflexivität folgen direkt aus Definition 1.

Theorem
Wenn $x = \langle X,P,N,O\rangle \in ARG$ und S eine Strukturmatrix ist, dann gilt für alle $x, y \in X$:
(1) $S(x,y) = S(y,x)$
(2) $S(x,x) = p^*$

Der Beweis ist trivial.

7.1.2 Symbolic Psycho-Logic und Balancegenerierung

In Anlehnung an das Modell von Heider werden Regeln aufgestellt, die es erlauben, aus bestehenden Kognitionen neue Kognitionen herzuleiten. Hierzu werden zwei existierende Kognitionen mit einem Element, das sie gemeinsam haben, kombiniert. Insgesamt geben die Autoren die folgenden sieben Regeln an. Die Regeln (1)-(3) entsprechen dabei den balancierten Heider-Triaden (a)-(d) von Abb. 6.2 (S.120).

Regel 1: aPb und bPc impliziert aPc
Regel 2: aPb und bNc impliziert aNc
Regel 3: aNb und bNc impliziert aPc
Regel 4: Wenn aPc und aNc beide impliziert sind oder wenn die eine anfänglich galt und die andere impliziert wurde, dann gilt aAc
Regel 5: aAc und cPd impliziert aAd
Regel 6: aAc und cNd impliziert aAd
Regel 7: aAc und cAd impliziert aAd

Notwendige Voraussetzung für die Anwendung der Regeln ist, daß das Individuum motiviert ist, über das Thema nachzudenken. Als Aktivierungsanreize dienen insbesondere:

(1) Druck, um eine Entscheidung zu einem bestimmten Thema zu erreichen;
(2) Sozial abgeleitete Bedürfnisse, um über ein Thema informiert zu scheinen, andere zu beeindrucken etc;
(3) Relevanz des Themas für Bedürfnisse, Konflikte etc;
(4) Ein "allgemeiner kognitiver Stil" des Individuums, so daß Nachdenken per se befriedigend ist.

Angenommen, ein Individuum hat einen Anreiz, über ein Thema nachzudenken und

es besitzt bereits bestimmte relevante Kognitionen. Durch Anwendung der Regeln entdeckt es neue Kognitionen. Diese neuen Kognitionen können kompatibel sein mit bestehenden oder nicht. Betrachten wir hierzu das folgende Beispiel:

Bush (a) ist an einem Sturz (N) Husseins (b) interessiert.
Hussein (b) bekämpft (N) Kurden (c).
(Regel 3): Bush (a) kümmert sich um (P) Kurden (c)

Die Theorie postuliert, daß das Individuum durch Nachdenken die neue Kognition nach Anwendung von Regel 3 entdeckt. Angenommen, aus der Sicht eines Individuums bestand zwischen Bush und den Kurden ursprünglich eine negative Relation (aNc). Das Individuum wäre dann nach Anwendung von Regel 3 konfrontiert mit den widersprüchlichen Relationen aNc und aPc, woraus mit Regel 5 aAc folgt.

An dem Beispiel läßt sich verdeutlichen, daß die Folgerungen, die sich mit diesen Regeln ergeben, nicht notwendigerweise logisch sind. Vielmehr werden die Regeln (1)-(7) offensichtlich als empirische Gesetzmäßigkeiten und damit als Hypothese verstanden, daß Personen im Alltagsleben nicht logisch, sondern "psychologisch schließen". Entsprechend wird das kleine Regelwerk als Symbolic Psycho-Logic bezeichnet: "Such 'reasoning' would mortify a logician, yet it can be found in much this form inside millions of heads. Thus we speak of the formal system as psycho-logic rather than as logic" (Abelson/Rosenberg 1958: 5).
Definition 3 baut die obengenannten Regeln in die AR-Graphen ein.

Definition 3
Wenn $x = \langle X,P,N,O \rangle \in ARG$, dann erfüllt x die Psycho-Logik gdw für alle $x,y,z \in X$ gilt:
(1) wenn xPy und yPz dann xPz
(2) wenn xPy und yNz dann xNz
(3) wenn xNy und yNz dann xPz
(4) wenn xPy und xNy dann xAy
(5) wenn xAy und yPz dann xAz
(6) wenn xAy und yNz dann xAz
(7) wenn xAy und yAz dann xAz

Bestimmte Strukturmatrizen sind durch die Eigenschaft charakterisiert, daß keine ambivalenten A-Relationen entdeckt bzw. abgeleitet werden können. Ist dies der Fall, so ist die kognitive Struktur balanciert. Ist die kognitive Struktur hingegen derart, daß eine oder mehrere ambivalente Relationen entdeckt bzw. abgeleitet werden, so ist die Struktur unbalanciert.

Definition 4
(1) Eine Strukturmatrix S ist unbalanciert gdw es *ein* $x,y \in X$ *gibt*, so daß gilt: xAy.
(2) Eine Strukturmatrix S ist balanciert gdw es *kein* $x,y \in X$ gibt, so daß gilt: xAy

Wie bei Heider wird angenommen, daß unbalancierte Strukturen instabil sind und ein Druck besteht, diese zu ändern. Hierzu werden drei Methoden genannt:

(1) Ändere eine oder mehrere der Relationen;
(2) Redefiniere, "differenziere" oder "isoliere" eines oder mehrere der Elemente;
(3) Hör auf nachzudenken.

Methode (3) ist selbsterklärend. Methode (2) bedeutet, ein Element z.B so in zwei Komponenten zu zerlegen, daß Balance wieder hergestellt ist. Hat z.B. Ego eine positive Einstellung zur Kirche und ein Freund von Ego eine negative, so ist diese Triade zunächst unbalanciert. Ego kann nun Balance dadurch erzeugen, daß er das Objekt Kirche in zwei Komponenten zerlegt, z.B. in die Komponente *Institution* Kirche, die sowohl Ego als auch der Freund ablehnt und in die Komponente *Religiösität*, der beide positiv gegenüberstehen. Abb. 7.1 veranschaulicht diesen Sachverhalt grafisch.

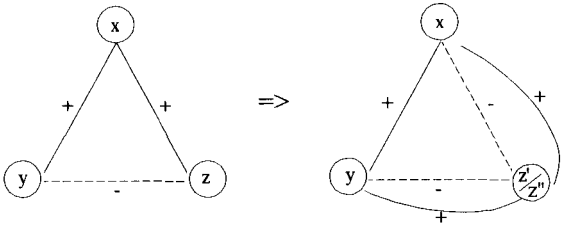


Abb. 7.1: Herstellung von Balance durch Zerlegung eines Objekts z in zwei Komponenten z' und z"

Das Modell behandelt jedoch nicht diese Möglichkeit der Differenzierung von Elementen, sondern beschränkt sich auf Methode (1), die *Änderung von Relationen*. Grundlegende Annahme bei der Relationenänderung ist dabei das *Ökonomieprinzip*. Nach diesem Prinzip wird eine unbalancierte Struktur derart in eine balancierte überführt, daß der *psychische Aufwand möglichst gering* ist, d.h. daß *möglichst wenige Relationen geändert werden müssen*. Vorausgesetzt wird, daß alle Relationen gleich resistent sind gegen einen Wandel.

Die Überführung unbalancierter in balancierte Strukturen gemäß dem Ökonomieprinzip basiert auf einem mathematischen Apparat, der hier nicht formal dargestellt wird (vgl. hierzu Abelson/Rosenberg 1958: 8-13). Wir stellen nur den endgültigen Algorithmus in der Terminologie der Autoren vor.

Nicht balancierte Strukturmatrizen sind durch einen mehr oder weniger hohen Grad an Unbalanciertheit gekennzeichnet, der "Komplexität von Unbalanciertheit" genannt wird. Die Komplexität von Unbalanciertheit wird definiert als die Minimalzahl von Relationenänderungen, die nötig sind, um Balance zu erreichen. Müssen z.B. zwei Relationen geändert werden, ist die Komplexität der Unbalanciertheit zwei. Das Modell bestimmt, welche Relationenänderungen diese Minimalmenge konstituieren. Falls es alternative Minimalmengen gibt, identifiziert das Modell alle. Auszuführen ist folgender Algorithmus, den wir zunächst informell wiedergeben:

(1) Schreibe die Strukturmatrix nieder.
(2) Suche die Spalte (Zeile), in der die Zahl der n* - p* maximal wird, exklusiv des p* in der Hauptdiagonale.

(3) Ändere alle n* in p* und alle p* in n*, sowohl in der Zeile und der korrespondierenden Spalte. Das Diagonalelement bleibt ungeändert, o*-Eingänge werden ignoriert.
(4) Wiederhole Schritt (2) und (3) für eine neue Zeile (Spalte) solange, bis die Anzahl der n* nicht weiter reduziert werden kann.

Die Matrixeingänge, in denen die n* erscheinen, identifizieren diese Minimalmenge. Wenn die Minimalzahl auf mehr als eine Möglichkeit gebildet werden kann, sind die so identifizierten Mengen alternative Lösungen. Ist die Minimalzahl 0, ist die Struktur balanciert.

Das Vorgehen sei an dem obigen Beispiel illustriert. Tab. 7.3 zeigt nochmals die Ausgangsstrukturmatrix, mit der Anzahl der n* und p* (ohne Diagonalelement) am Ende jeder Zeile.

Tab. 7.3: Ausgangsstrukturmatrix

	Ego	The hon. stud.	Rep. Chea	Feel. trust	Chea by a few	Honor Syst.	#n*	#p*	
Ego	p*	p*	n*	p*	n*	o*	2	2	
The honest student	p*	p*	n*	p*	o*	p*	1	3	
Reporting cheaters	n*	n*	p*	o*	n*	p*	3	1	
Feeling trusted	p*	p*	o*	p*	o*	p*	0	3	
Cheating by a few	n*	o*	n*	o*	p*	n*	3	0	⇐
An honor system	o*	p*	p*	p*	n*	p*	1	3	
					⇑				

In Zeile 5 wird die Zahl der n*-p* mit drei n* maximal, so daß die Änderungsoperationen in Zeile 5 und Spalte 5 durchzuführen sind. Die Umwandlung von n* in p* und p* in n* - mit Ausnahme der Diagonalzelle - ergibt folgendes Ergebnis.

Tab. 7.4: Ergebnis der ersten Modifikation

	1	2	3	4	5	6	#n*	#p*	
1	p*	p*	n*	p*	p*	o*	1	3	
2	p*	p*	n*	p*	o*	p*	1	3	
3	n*	n*	p*	o*	p*	p*	2	2	⇐
4	p*	p*	o*	p*	o*	p*	0	3	
5	p*	o*	p*	o*	p*	p*	0	3	
6	o*	p*	p*	p*	p*	p*	0	4	
			⇑						

Zwei n* verbleiben in Zeile 3, denen - das Diagonalelement ausgenommen - zwei p* gegenüberstehen. Die Änderungsoperationen in Zeile 3, Spalte 3 führen zu Tab. 7.5.

Tab. 7.5: Ergebnis der zweiten Modifikation

	1	2	3	4	5	6	#n*	#p*
1	p*	p*	p*	p*	p*	o*	0	4
2	p*	p*	p*	p*	o*	p*	0	4
3	p*	p*	p*	o*	n*	n*	2	2
4	p*	p*	o*	p*	o*	p*	0	3
5	p*	o*	n*	o*	p*	p*	1	2
6	o*	p*	n*	p*	p*	p*	1	3

Die Anzahl der n* ändert sich nicht und bleibt bei zwei. Es ist evident, daß die Zahl der n* nicht weiter reduziert werden kann, so daß die Strukturmatrix die Komplexität zwei hat und es zwei alternative Minimallösungen für das Balanceproblem gibt.

Die erste Lösung lautet, daß die Relationen zwischen den Elementen 3 und 1 sowie 3 und 2 geändert werden müssen. Wären sie anders, so bestünde Gleichgewicht. Die Relationen zwischen Objekt 3 und 1 sowie 3 und 2 müssen folglich von einer N- in eine P- oder O-Relation transformiert werden, damit Balance erreicht wird.
Die zweite - alternative - Lösung besteht darin, daß das Subjekt die Relationen zwischen den Elementen 3 und 5 sowie 3 und 6 zu P bzw. O korrigiert.

Unter der Annahme, daß alle Relationen gleich resistent gegen Änderungen sind, sind die beiden Alternativlösungen die einfachste Art, Gleichgewicht zu erreichen.

Zur formalen Festlegung dieses Verfahrens definieren wir ein zweistelliges Prädikat "... ist eine Verbesserung von ...". Im Definiens dieser Definition wird eine neue Strukturmatrix S* eingeführt, die eine Verbesserung der alten Strukturmatrix S sein soll. Die Überführung von S in S* nach dem eben genannten Verfahren erfolgt in Definition 5-1 bis 5-3.

Definition 5

Sei $x \in ARG$, S die Strukturmatrix von x und S unbalanciert.

S* *ist eine Verbesserung von* S gdw

$S^*: X \times X \rightarrow \{p^*, n^*, o^*\}$ und es gibt $a \in X$ so daß

(1) Für alle $b \in X$: $card(\{c \in X \mid S(a,c)=n^*\}) - card(\{c \in X \mid S(a,c)=p^* \wedge c \neq a\})$
$\geq card(\{c \in X \mid S(b,c)=n^*\}) - card(\{c \in X \mid S(b,c)=p^* \wedge c \neq b\})$

(2) Für alle $c \in X$: wenn $a \neq c$ dann
wenn $S(a,c) = n^*$ dann $S^*(a,c) = p^*$
wenn $S(a,c) = p^*$ dann $S^*(a,c) = n^*$
wenn $S(c,a) = n^*$ dann $S^*(c,a) = p^*$
wenn $S(c,a) = p^*$ dann $S^*(c,a) = n^*$

(3) Für alle c,d mit $c \neq a$ und $d \neq a$: $S^*(c,d) = S(c,d)$

7.1.3 Theoriekern und intendierte Anwendungen

Zu ergänzen bleibt nun noch der formale Theoriekern der Balancetheorie von Abelson-Rosenberg, also die Definition des potentiellen Modells M_p und des Modells M. Analog zu Heider werden im potentiellen Modell die Grundbegriffe um die Zeit und die Kleiner-Relation erweitert.

Definition 6
x ist ein *potentielles Modell* der Abelson-Rosenberg-Theorie ($x \in M_p(ART)$) gdw es X, T, <, P, N, O gibt, so daß gilt:
(1) $x = \langle X,T,<,P,N,O\rangle$
(2) X ist eine endliche, nicht-leere Menge
(3) $\langle T,< \rangle$ ist eine endliche, lineare Ordnung
(4) $P: T \rightarrow Pot(X \times X)$, $N: \rightarrow Pot(X \times X)$, $O: \rightarrow Pot(X \times X)$
(5) Für alle $t \in T$: $\langle X, P(t),N(t),O(t)\rangle \in ARG$

Die Definition des potentiellen Modells erfolgt völlig parallel zu Heider und bedarf keines Kommentars. Die Bemerkungen zu T-theoretischen Termen gelten hier ebenfalls analog zu HT: in ART gibt es keine Terme, deren Werte nicht empirisch bestimmt werden könnten; damit existieren keine AR-theoretischen Begriffe.
Definition 7 legt abkürzende Symbole fest und Definition 8 schließlich das Modell der Theorie mit dem "eigentlich inhaltlichen" Axiom (3).

Definition 7
(1) $x(t) := \langle X, P(t),N(t),O(t)\rangle$
(2) $S_t := S = S_{x(t)}$

Definition 8
x ist ein *Modell* der Abelson-Rosenberg-Theorie ($x \in M(ART)$) gdw es X, T, <, P, N, O gibt, so daß
(1) $x = \langle X,T,<,P,N,O\rangle$
(2) $x \in M_p$
(3) Für alle $t \in T$ gibt es $t' \in T$ so daß gilt: wenn $t < t'$ und S_t unbalanciert ist, dann ist $S_{t'}$ balanciert und durch eine Folge von Verbesserungen aus S_t entstanden

Axiom (3) von Definition 8 drückt das Gesetz von ART aus, nach der es zu unbalancierten Strukturmatrizen für beliebige Zeitpunkte t einen Zeitpunkt t' gibt, bei dem die Matrix balanciert ist und die balancierte Matrix Ergebnis von Verbesserungen der unbalancierten Matrix ist. Alle potentiellen Modelle, die Axiom (3) von Definition 8 erfüllen, sind Modelle der Abelson-Rosenberg-Theorie.

Die Menge der intendierten Anwendungen von ART besteht, wie eingangs erwähnt, aus Einstellungsräumen, also kognitiven Systemen, deren paarweise Relationen Bewertungen zulassen (Glaubens- oder Überzeugungssysteme). Die ursprüng-

lich genannte Beispielmenge $I_0 \subseteq I$ beschränkt sich auf den Einstellungsraum "Having an honor system at Yale". Weitere intendierte Beispielsysteme lassen sich leicht angeben, wie etwa Themen zu Religion, Politik, Gesundheit etc. Sicherlich von Abelson und Rosenberg nicht-intendierte Anwendungen sind reine Wissenssysteme, wie sie z.B. Theorien darstellen, das Wissen, wie man mathematische Beweise führt oder einen Computer repariert. Die Theorie ist sicherlich auch nicht intendiert für nicht-kognitive Systeme, was nicht heißt, daß es vielleicht doch solche Anwendungsmöglichkeiten gibt.

Irle (1975) verweist darauf, daß die Theorie trotz ihrer bestechenden Einfachheit und Generalität nur in spärlichem Maß empirischen Überprüfungen unterzogen wurde. Zwei Experimente von Rosenberg/Abelson (1960) bestätigen, daß diejenige Alternative zur Herstellung kognitiven Gleichgewichts bevorzugt wird, welche den Aufwand minimiert. Burnstein (1967) legte Versuchspersonen balancierte und unbalancierte Einstellungsräume vor mit der Frage, ob und in welcher Weise sich die Konfigurationen ändern würden. Erwartungsgemäß nahmen die Versuchspersonen häufiger an, daß sich unbalancierte Kognitionssysteme in balancierte ändern würden als umgekehrt. Es wurden jedoch solche Konfigurationen häufiger vorhergesagt, welche eine Vermehrung positiver anstelle einer Verminderung negativer Beziehungen beinhalteten. Außerdem wurden in den Vorhersagen balancierte Strukturen bevorzugt, die eher Änderungen persönlicher Beziehungen als Änderung der Beziehungen zu Objekten/Ereignissen in der sozialen Umwelt beinhalteten. Irle (1975) vermutet daher, daß der Verbesserungs-Algorithmus, der auf einer Minimimierung des Änderungsaufwands beruht, durch "theorie-externe Zusatzhypothesen" modifiziert werden müßte.

7.2 Wissenbasierte Modellierung

Wir verzichten bei den folgenden Modellen auf die strukturalistische Maschinendarstellung der Prädikate und implementieren die Theorien als Datengenerierungs-Modelle. Das Computermodell der Balancetheorie von Abelson/Rosenberg soll folgende Funktionen beinhalten:

- Daten einlesen und die Relationen klassifizieren;
- Anwendung der Psycho-Logik-Regeln von Definition 3 sowie Definition 4 und Generierung neuer Daten;
- Transformation der Daten gemäß dem obengenannten Algorithmus.

7.2.1 Datenrepräsentation und -generierung

Wir haben im Heider-Modell die P- und N-Relationen genauso wie in der Prädikatenlogik repräsentiert, nämlich als 2-stellige Prädikate mit dem Relationennamen als Funktor und den Objektnamen als Argumente:

```
relation(objekt1,objekt2).
```

Diese Darstellung hat einige Nachteile. Zum ersten läßt sich der Relationenname

nicht ohne weiteres instantiieren, was insbesondere im Hinblick auf die Erklärungskomponente ungünstig ist. Zum zweiten wollen wir auch an irgendeiner Argumentstelle festhalten, *warum* der Fakt gültig ist, ohne die Relationsprädikate künstlich um eine Stelle zu erweitern. Wir führen deshalb ein allgemeines Prädikat `faktum(x,y)` ein, an dessen einer Argumentstelle ein Fakt der Wissensbasis steht und an dessen anderer Stelle die Ursache seines Bestehens erscheint. Die eigentlichen Daten befinden sich an der zweiten Argumentstelle `y`, während an der ersten Stelle `x` die Information steht, ob der Fakt empirisch oder abgeleitet ist. Ist es ein Grundfaktum, erscheint das Atom `empir` an erster Argumentstelle. Ist es abgeleitet, so besteht `x` aus einer Liste, welche die zur Ableitung benutzte Regelnummer und die Regelbedingungen enthält.

Das Prädikat `faktum/2` hat somit im Fall, daß das zweite Argument ein Grundfakt ist, die Ausprägung

```
faktum(empir,rel(objekt1,objekte2)).
```

und im Fall, daß das zweite Argument ein abgeleiteter Fakt ist, die Ausprägung

```
faktum([regelnr,beding1,...],rel(objekt1,objekt2)).
```

Die Erhebung der Daten muß, anders als im Fall der strukturalistischen Prädikate, *vollständig* durchgeführt werden. In den strukturalistischen Modellprädikaten konnte die Datenbasis unvollständig sein mit der Konsequenz, daß die entsprechenden PROLOG-Prädikate bei der Anfrage fehlschlugen. Bei Datengenerierungs-Modellen müssen wir den Benutzer zwingen, die Fakten vollständig einzugeben. Wir halten uns bei der Datenerhebung mit den folgenden vier Schritten eng an das Vorgehen von Abelson/Rosenberg.

1. *Objektmenge X sammeln*
   Zunächst werden die Objekte des Einstellungsraums in der Objektliste `X` gesammelt, im Einstellungsraum "Golfkrieg" könnte dies z.B. sein:
   ```
   X := [bush, hussein, kurden, israel].
   ```
   Zum Einlesen der Objekte in die Liste verwenden wir das Prädikat `read_in/2`, dessen Funktionsweise z.B. in Clocksin/Mellish (1987: 102-104) oder Schnupp (1986: 175-179) beschrieben ist und das auch schon bei Heider benutzt wurde.
2. *Objekt ego hinzufügen*
   Das Objekt `ego` wird in die Liste `X` eingefügt ("geconst", in Anlehnung an die Funktion "cons" in LISP):
   ```
   X := {ego} ∪ X -> [ego, bush, hussein, kurden, israel].
   ```
   Beide Schritte, Objektmenge einlesen und das Atom `ego` hinzufügen, übernimmt die Prozedur `lies_objekte/0` (vgl. Anhang).
3. *Kreuzprodukt bilden*
   ```
   X x X = [[ego,ego],[ego,bush], [ego,hussein], ... ,
   [israel,israel]]
   ```
   Die Bildung des Kreuzproduktes übernimmt `generiere_kreuz/0` (vgl. Anhang). Sowohl das Kreuzprodukt als auch die Objektliste werden in die

Datenbasis eingetragen.

4. *Relationen zwischen $X \times X$ erheben*
   Aufgrund der Reflexivität von P und Symmetrie von P, N und O müssen nicht alle Elemente der Kreuzproduktliste abgefragt werden. Beispielsweise braucht die Relation `[bush, bush]` nicht ermittelt zu werden. Ebenso muß die Beziehung `[bush,ego]` nicht erhoben werden, wenn die Relation zwischen `[ego,bush]` schon festgestellt wurde.

Das Abfragen der Relationen wird angestoßen von `lies_relationen/0`, welches die Kreuzproduktliste instantiiert und dem Prädikat `lies_rel/1` übergibt. Dieses arbeitet die Kreuzproduktliste in folgender Reihenfolge rekursiv ab:

1. Ist die Liste leer? Dann ist die Operation beendet.
2. Ist das Listenelement vom Typ `[a,a]`? Dann ist gemäß Definiton 1, Axiom (7) die Relation positiv und braucht nicht abgefragt zu werden. `faktum(empir,p(a,a))` wird der Wissensbasis hinzugefügt.
3. Ist das Listenelement vom Typ `[a,b]` mit $a \neq b$ und wurde `[b,a]` bereits abgefragt? Dann braucht `[a,b]` nicht mehr geprüft zu werden und `faktum(empir,rel(a,b))` wird mit der Relation, die zwischen b und a bestand, der Wissensbasis hinzugefügt.
4. Ist das Listenelement vom Typ `[a,b]` mit $a \neq b$, und wurde `[b,a]` noch nicht abgefragt? Dann frage den Benutzer, welche Relation zwischen den Elementen besteht. Der Benutzer kann eine *beliebige* Relation eingeben. Der Einfachheit halber setzen wir voraus, daß für Relationen lediglich PROLOG-Atome zugelassen sind. Die Relation wird anhand eines kleinen Lexikons klassifiziert in die Modellrelationen P, N und O. Findet das System keinen entsprechenden Eintrag im Lexikon, stellt es an den Benutzer eine entsprechende Frage. Die Kognition wird in die Synonymen-Liste eingetragen.

```
lies_rel([]).                          % (1) Liste leer? -> fertig

lies_rel([[A,A]|Rest]) :-              % (2) Listenelement vom Typ [a,a]?
   assertz(faktum(empir,p(A,A))),      %     -> füge p(a,a) hinzu
   lies_rel(Rest).                     %     lies die Restliste ein

lies_rel([[A,B]|Rest]) :-              % (3) Wurde [a,b] bereits gefragt,
   symmetrie(Liste),                   %     d.h. ist [a,b] Element der
   member([X,A,B],Liste),              %     aktuellen Symmetrieliste
   F=..[X,A,B],
   assertz(faktum(empir,F)),           %     -> füge Relation hinzu
   lies_rel(Rest).                     %     lies die Restliste ein

lies_rel([[A,B]|Rest]) :-              % (4) Frage den Benutzer
   schreib_liste(['Relation',A,B]),
   read(Relation),                     %     lies die Relation
   klassifiziere(Relation,X),          %     klassifiziere Relation
   F=..[X,A,B],
   assertz(faktum(empir,F)),           %     trage Relation ein
   trage_ein_sym([X,B,A]),!,nl,        %     trage Rel. in Synonym.liste
   lies_rel(Rest).                     %     lies die Restliste ein
```

Alle Einleseprozeduren werden durch das Prädikat `lesen/0` in der entsprechenden Reihenfolge aufgerufen. Die so erhobenen Daten entsprechen den Forderungen von Definition 1: es gibt eine Objektmenge X und paarweise disjunkte, symmetrische Relationen P, N, O auf $X \times X$ mit xPx für alle $x \in X$.

7.2.2 Psycho-Logik-Regeln und Balancegenerierungs-Algorithmus

Die eigentlichen Regeln des Programms, mit denen Ableitungen und Erklärungen möglich sind, bilden die Axiome von Definition 3 und 4. Die Psycho-Logik-Regeln von Definition 3 erlauben die Ableitung neuer Kognitionen aus bestehenden, und die beiden Axiome von Definition 4 klassifizieren einen Einstellungsraum als gleichgewichtig und ungleichgewichtig.

Wir schreiben uns hierzu zwei einfache eigene Regelinterpreter, welche die Abarbeitung der Regeln übernehmen. In der ersten Version soll der Regelinterpreter dafür sorgen, daß *alle* möglichen Regeln von Definition 3 auf die Daten angewendet werden. Die Konklusionen sollen zusammen mit der Ursache ihres Bestehens der Faktenbasis hinzugefügt werden. Mit der zweiten Version soll es möglich sein, nur einzelne interessierende Fakten zu deduzieren.

Der Regelinterpreter in der ersten Version wird angestoßen durch Aufruf des Prädikats `leite_ab/0`, welches der Kopf der folgenden Regel ist.

```
leite_ab :-
   regel(Nr,Konkl,Beding),          % Instantiiere eine Regel, deren
   not(faktum(_,Konkl)),            % Konkl. noch nicht i.d. Faktenbasis ist
   assertz(faktum([Nr|Beding],Konkl)), % füge Konkl. d. Fakt.basis hinzu
   schreib_liste([Konkl,'abgeleitet mit Regelnr ',Nr]),
   fail.                                  % bringe leite_ab zum Scheitern
```

`leite_ab/0` kann als Meta-Regel betrachtet werden, mit der versucht wird, eine Regel durch Instantiierung eines entsprechenden `regel`-Prädikats anzuwenden. Ist die Anwendung einer Regel erfolgreich, kann die Konklusion abgeleitet und der Datenbasis hinzugefügt werden. Die zweite Clause verhindert, daß die Folgerung mehrmals in die Faktenbasis eingetragen wird. Ist die Deduktion nicht in der Wissensbasis vorhanden, wird die aktuelle Faktenbasis durch Eintrag an die zweite Argumentstelle von `faktum/2` erweitert. An der ersten Argumentstelle erscheint eine Liste, bestehend aus der Regelnummer und den Bedingungen der Regel, die zur Ableitung führten. Das `fail` am Ende der `leite_ab`-Regel bewirkt ein Backtracking zum Prädikat `regel/3`, so daß alle potentiell anwendbaren Regeln überprüft werden.

Das `leite_ab`-Prädikat in der zweiten Version des Regelinterpreters ist 1-stellig und erwartet als Argument ein potentiell gültiges Faktum. Dabei bedienen wir uns der üblichen Vorgehensweise bei Expertensystemen (vgl. z.B. Savory 1988, Schnupp/Nguyen Huu 1987). Angenommen, es soll gezeigt werden, daß der Fakt x gültig ist, dann soll der Regelinterpreter

(1) prüfen, ob x bereits in der Faktenbasis steht; wenn dies nicht der Fall ist, dann
(2) prüfen, ob x mit einer Regel abgeleitet werden kann; wenn dies nicht der Fall ist, dann

(3) den Benutzer nach der Gültigkeit von x fragen.
Schritt (3) entfällt, da wir davon ausgehen, daß die Grundrelationen als die einzigen nicht ableitbaren Fakten vollständig in der Datenbasis stehen.
`leite_ab(Faktum)` ist also erfolgreich, wenn entweder `Faktum` ein Faktum ist oder eine Regel angewendet werden kann.

```
leite_ab(Konkl) :-                          % Konkl ist true wenn
   faktum(_,Konkl).                         % Konkl ein Faktum ist

leite_ab(Konkl) :-                          % Konkl ist true wenn
   regel(Nr,Konkl,Beding),                  % es eine Regel gibt und
   not faktum(_,Konkl),                     % Konkl noch kein Fakt ist und
   assertz(faktum([Nr|Beding],Konkl)),      % ...
   schreib_liste(
   [Konkl,' abgeleitet mit Regelnr ',Nr]).
```

Die eigentlichen Regeln von Definition 3 und 4 sind in folgenden acht PROLOG-Clausen codiert.

```
regel(1,p(A,C),[p(A,B),p(B,C)]) :-
   faktum(empir,p(A,B)),
   faktum(empir,p(B,C)).

regel(2,n(A,C),[p(A,B),n(B,C)]) :-
   faktum(empir,p(A,B)),
   faktum(empir,n(B,C)).

regel(3,p(A,C),[n(A,B),n(B,C)]) :-
   faktum(empir,n(A,B)),
   faktum(empir,n(B,C)).

regel(4,a(A,C),[p(A,C),n(A,C)]) :-
   faktum([_|_],p(A,C)),
   faktum([_|_],n(A,C)).

regel(5,a(A,C),[p(A,C),n(A,C)]) :-
   faktum(empir,p(A,C)),
   faktum([_|_],n(A,C)).

regel(6,a(A,C),[p(A,C),n(A,C)]) :-
   faktum([_|_],p(A,C)),
   faktum(empir,n(A,C)).

regel(7,ungleichgewicht,[a(A,B)]) :-
   faktum([_|_],a(A,B)).

regel(8,gleichgewicht,['nicht es gibt x,y: a(x,y)']) :-
   not faktum([_|_],a(_,_)).
```

Regel 1 kann beispielsweise so gelesen werden: Wenn empirisch zwischen A und B eine positive Relation besteht und zwischen B und C ebenfalls eine positive Relation herrscht, dann liegt auch zwischen A und C eine positive Relation vor. Die Regeln 4-6 beziehen sich auf die im Originalbeitrag gemachte Unterscheidung von abgeleiteten und empirischen Daten. Regel 4 besagt z.B. folgendes: wenn `p(A,C)` und `n(A,C)` beide abgeleitet wurden (erkennbar an der Liste an erster Argumentstelle), dann gilt

die ambivalente Relation `a(A,C)`. Regel 7 besagt, daß Ungleichgewicht vorliegt, wenn es gelingt, eine A-Relation abzuleiten und Regel 8 entsprechend, daß Gleichgewicht besteht, wenn ein Beweisversuch für `a(_,_)` scheitert. `ungleichgewicht` und `gleichgewicht` werden hierbei im Unterschied zu Definition 4 als 0-stellige Prädikate eingeführt.

Die Regeln 5-7 aus Definition 3 fehlen. Der Grund ist, daß bei Anwendung dieser Regeln alle Objektpaare von A-Relationen "infiziert" würden, was wenig sinnvoll erscheint, da schließlich zwischen allen Objekten nur mehr A-Relationen bestünden. Die Bildung der Strukturmatrix (Definition 2) wird durch `generiere_matrix/0` angestoßen und der Balancegenerierungs-Algorithmus (Definition 5) durch `verbessere_matrix/0`. Der Algorithmus wurde oben beschrieben, die Implementierung kann dem Anhang entnommen werden. Bei diesem algorithmischen Programmteil entfällt die deklarative Sicht und damit der Vorteil von PROLOG vollständig. Die Verwendung von PROLOG unterscheidet sich hier in keinem Punkt von einer konventionellen Sprache, und die Implementierung des Algorithmus wäre mit einer konventionellen Sprache genauso, wenn nicht eleganter, durchführbar. Jeder Aufruf von `verbessere_matrix/0` verbessert die Matrix im Sinn von Definition 5. Läßt sich eine Matrix nicht weiter vervollkommnen, scheitert der Aufruf des Algorithmus. Der Test mit dem Originalbeispiel von Abelson/Rosenberg ergab die gleichen Verbesserungsfolgen der Strukturmatrix und terminiert an der gleichen Stelle.

7.2.3 Erklärungskomponente

Eine zentrale Forderung an ein wissensbasiertes System war, daß es seine Schlüsse erklären und begründen kann. Das Modell soll deshalb dem Benutzer anzeigen, wie es zu seinen Ableitungen gekommen ist. Im einfachsten Fall soll es z.B. durch die Anfrage `wie(a(objekt1, objekt2))` angeben, wie es darauf kommt, daß zwischen `objekt1` und `objekt2` eine Ambivalenz-Relation besteht. Eine Erklärungsmöglichkeit für Schlußfolgerungen ist im Gegensatz zum Heider-Modell in diesem Modell insofern geboten, als eine Vielzahl von nur mehr schwer überblickbaren Ableitungen vorkommen kann. Wir implementieren zunächst folgende einfache Erklärungskomponente, welche die Begründungen direkt dem Programmcode entnimmt ("direkte Erklärungen", vgl. Kap. 4.1.2, S.69). Vorbild für das hier und insbesondere in Kap. 9.4.4 realisierte Modul ist Schnupp/Nguyen Huu (1987), Kap. 5 "Dialogführung und Erklärungskomponenten".

```
wie(Konkl) :-
   faktum([Nr|Bed_liste],Konkl),
   write('\n\n Relationen: ' ),
   schreib_liste(Bed_liste),
   gib_regel_aus(Nr),
   write('Also: '),
   write(Konkl).
```

```
gib_regel_aus(Nr) :-
   clause(regel(Nr,Konkl,_), Ursachen),
   schreib_liste(['Regel',Nr,') ']),
   write('WENN '),
   schreibe_ursachen(Ursachen),
   write(' DANN '),
   schreibe_konklusion(Konkl).
```

Die Regel `wie(Konkl)` erwartet als Argument eine (abgeleitete) Kognition oder einen Statuszustand (`gleichgewicht/ungleichgewicht`). Das erste Prädikat `faktum` im Regelkörper versucht über die Variable `Konkl` ein Matching mit einem entsprechenden `faktum`-Prädikat in der Wissensbasis. Gelingt das Matching, wird das erste Argument von `faktum/2` mit der Bedingungsliste der Regel instantiiert. In diesem Fall werden die Bedingungen, welche zur Ableitung führten, mit `schreib_liste/1` ausgegeben, die Clause `gib_aus_regel(Nr)` bewirkt den Regeloutput, und schließlich wird die Konklusion nochmals angezeigt.

Das folgende Prädikat `wie/0` bewirkt eine Schlußrechtfertigung für *alle* abgeleiteten Fakten.

```
wie :-
   findall(Konkl,faktum([Nr|_],Konkl),K_Liste),
   konkl_liste(K_Liste).

konkl_liste([]).
konkl_liste([Kopf|Rest]) :-
   wie(Kopf),
   konkl_liste(Rest).
```

Hierzu werden mit dem Built-in-Prädikat `findall/3` alle deduzierten Konklusionen in der Liste `K_Liste` gesammelt. Das Prädikat `konkl_liste` arbeitet diese Liste rekursiv ab, wobei im Regelkörper die `wie`-Regel für jeweils ein Listenelement (d.h. eine Konklusion) aufgerufen wird.
Das Prädikat `regeln/0` schließlich führt dazu, daß *alle* Regeln ausgegeben werden.

```
regeln :-
   alle_regeln(1).

alle_regeln(9) :-
   write('\nOK - All rules').

alle_regeln(N) :-
   gib_regel_aus(N),
   N1 is N+1,
   alle_regeln(N1).
```

Hilfsprädikate wie das Prädikat `objekte/0` (listet alle Objekte auf), `fakten/0` (listet alle Fakten auf) etc. sind für das Modell substantiell wenig interessant und werden deshalb hier nicht vorgestellt. Deren Arbeitsweise kann dem Programmcode im Anhang entnommen werden. Der Aufruf von `hilfe` gibt eine Überblick über alle sinnvoll verwendbaren Prädikate.

7.2.4 Modelldialoge

Modelldialog 1: Einstellungsraum Golfkrieg

Der folgende Programmdialog basiert auf empirischen Daten einer Versuchsperson zum Thema Golfkrieg mit einer 4-elementigen Objektliste. Benutzereingaben sind wieder fett gedruckt.

```
?- hilfe.

******** BALANCETHEORIE VON ABELSON/ROSENBERG (ART) *************

Moegliche Eingaben:

DATENBASIS AENDERN
- Daten eingeben                          -> lesen.
- Alte Wissensbasis voellig loeschen      -> neu.
- Nur Schlussfolgerungen loeschen         -> entferne(schluesse).
- Matrix loeschen                         -> entferne(matrix).
- Einzelnen Fakt entfernen                -> entferne(fakt).
- Einzelnen Fakt hinzunehmen              -> fuege_hinzu(fakt).
FAKTEN UND SCHLUESSE
- Alle Schluesse generieren               -> leite_ab.
- Einzelne Fakten beweisen                -> leite_ab(fakt).
- Ueberblick ueber gueltige Fakten        -> fakten.
- Ueberblick ueber vorhandene Objekte     -> objekte.
- Widerspruechliche Fakten                -> zeige_wiedersprueche.
- Schlussrechtfertigung fuer Fakt         -> wie(fakt).
- Schlussrechtfertigung fuer alle Fakten -> wie.
MATRIZENOPERATIONEN/BALANCEGENERIERUNG
- Strukturmatrix erzeugen                 -> generiere_matrix.
- Aktuelle Matrix ausgeben                -> matrix.
- Folge verbesserter Matrizen erzeugen    -> verbessere_matrix.

yes

?- lesen.

Eingabe der Objekte > bush hussein kurden israel.
Relation ego bush
positiv.
p(ego,bush)
Relation ego hussein
negativ.
n(ego,hussein)
Relation ego kurden
positiv.
p(ego,kurden)
Relation ego israel
positiv.
p(ego,israel)
Relation bush hussein
hates.
n(bush,hussein)
Relation bush kurden
positiv.
p(bush,kurden)
Relation bush israel
positiv.
```

```
p(bush,israel)
Relation hussein kurden
negativ.
n(hussein,kurden)
Relation hussein israel
negativ.
n(hussein,israel)
Relation kurden israel
positiv.
p(kurden,israel)
yes

?- objekte.

ego bush hussein kurden israel

yes

?- leite_ab(ungleichgewicht).

no

?- leite_ab.

gleichgewicht  abgeleitet mit Regelnr  8

yes

?- wie.

Empirische Relationen: nicht es gibt x,y: a(x,y)
Regel 8)
WENN  NOT a(x,y) (ABGELEITET) DANN gleichgewicht

Also: gleichgewicht

yes
```

Kommentar: Der Versuch, Ungleichgewicht zu beweisen, schlug ebenso fehl wie die Ableitung neuer Kognitionen. Das einzige, mit `leite_ab` produzierbare Deduktionsergebnis ist, daß das System balanciert ist.

```
?- fakten.

Empirisch:
p(ego,ego)
p(ego,bush)
n(ego,hussein)
p(ego,kurden)
p(ego,israel)
p(bush,ego)
p(bush,bush)
n(bush,hussein)
p(bush,kurden)
p(bush,israel)
n(hussein,ego)
n(hussein,bush)
p(hussein,hussein)
```

```
n(hussein,kurden)
n(hussein,israel)
p(kurden,ego)
p(kurden,bush)
n(kurden,hussein)
p(kurden,kurden)
p(kurden,israel)
p(israel,ego)
p(israel,bush)
p(israel,bush)
n(israel,hussein)
p(israel,kurden)
p(israel,israel)

Abgeleitet:
gleichgewicht  8

yes

?- regeln.

Regel 1)
WENN p(x,y)  UND  p(y,z)  DANN p(x,z)
Regel 2)
WENN p(x,y)  UND  n(y,z)  DANN n(x,z)
Regel 3)
WENN n(x,y)  UND  n(y,z)  DANN p(x,z)
Regel 4)
WENN p(x,y) (ABGELEITET) UND n(x,y) (ABGELEITET) DANN a(x,y)
Regel 5)
WENN p(x,y)  UND n(x,y) (ABGELEITET)  DANN a(x,y)
Regel 6)
WENN p(x,y) (ABGELEITET) UND n(x,y)  DANN a(x,y)
Regel 7)
WENN a(x,y) (ABGELEITET) DANN ungleichgewicht
Regel 8)
WENN  NOT a(x,y) (ABGELEITET) DANN gleichgewicht

OK - Alle Regeln

yes

?- generiere_matrix.

yes

?- matrix.

1  [p p n p p]
2  [p p n p p]
3  [n n p n n]
4  [p p n p p]
5  [p p n p p]

yes
```

Modelldialog 2: Einstellungsraum Kirche

Das folgende, ebenfalls empirisch gewonnene Beispiel, zeigt einen unbalancierten Einstellungsraum.

```
?- lesen.

Eingabe der Objekte > kirche papst drewermann zoelibat pille.

Relation ego kirche
negativ.
n(ego,kirche)
Relation ego papst
negativ.
n(ego,papst)
Relation ego drewermann
positiv.
p(ego,drewermann)
Relation ego zoelibat
negativ.
n(ego,zoelibat)
Relation ego pille
positiv.
p(ego,pille)
Relation kirche papst
positiv.
p(kirche,papst)
Relation kirche drewermann
negativ.
n(kirche,drewermann)
Relation kirche zoelibat
positiv.
p(kirche,zoelibat)
Relation kirche pille
negativ.
n(kirche,pille)
Relation papst drewermann
negativ.
n(papst,drewermann)
Relation papst zoelibat
positiv.
p(papst,zoelibat)
Relation papst pille
negativ.
n(papst,pille)
Relation drewermann zoelibat
negativ.
n(drewermann,zoelibat)
Relation drewermann pille
negativ.
n(drewermann,pille)
Relation zoelibat pille
neutral.
o(zoelibat,pille)
yes
```

```
?- leite_ab.

p(drewermann,pille) abgeleitet mit Regelnr  1
n(ego,pille) abgeleitet mit Regelnr  2
n(ego,drewermann) abgeleitet mit Regelnr  2
n(zoelibat,pille) abgeleitet mit Regelnr  2
p(kirche,pille) abgeleitet mit Regelnr  3
p(kirche,drewermann) abgeleitet mit Regelnr  3
p(papst,pille) abgeleitet mit Regelnr  3
p(papst,drewermann) abgeleitet mit Regelnr  3
p(zoelibat,pille) abgeleitet mit Regelnr  3
a(zoelibat,pille) abgeleitet mit Regelnr  4
a(ego,drewermann) abgeleitet mit Regelnr  5
a(ego,pille) abgeleitet mit Regelnr  5
a(drewermann,pille) abgeleitet mit Regelnr  6
a(kirche,pille) abgeleitet mit Regelnr  6
a(kirche,drewermann) abgeleitet mit Regelnr  6
a(papst,pille) abgeleitet mit Regelnr  6
a(papst,drewermann) abgeleitet mit Regelnr  6
ungleichgewicht  7
no
```

Kommentar: Das Modell hat eine Menge neuer Kognitionen erzeugt, wovon acht ambivalent sind. Der Benutzer kann sich jede dieser Ableitungen begründen lassen, z.B.:

```
?- wie(a(kirche, drewermann)).

Relationen: p(kirche,drewermann) n(kirche,drewermann)
Regel 6)
WENN p(x,z) (ABGELEITET) UND n(x,z)   DANN a(x,z)
Also: a(kirche,drewermann)
yes
```

Kommentar: Da `p(kirche,drewermann)` offensichtlich auch abgeleitet wurde, ergibt die Anfrage nach dieser Kognition die endgültige Schlußkette:

```
?- wie(p(kirche, drewermann)).

Relationen: n(kirche,pille) n(pille,drewermann)
Regel 3)
WENN n(x,y)   UND n(y,z)   UND  DANN p(x,z)
Also: p(kirche,drewermann)
yes

?- wie(a(ego,pille)).

Relationen: p(ego,pille) n(ego,pille)
Regel 5)
WENN p(x,z)   UND n(x,z) (ABGELEITET) DANN a(x,z)
Also: a(ego,pille)
yes
```

```
?- wie(n(ego,pille)).

Relationen: p(ego,drewermann) n(drewermann,pille)
Regel 2)
WENN p(x,y)   UND n(y,z)   UND  DANN n(x,z)
Also: n(ego,pille)
yes

?- zeige_widersprueche.

p(ego,drewermann) (empir)   inkonsistent mit  n(ego,drewermann) (2)
p(ego,pille) (empir)   inkonsistent mit  n(ego,pille) (2)
n(kirche,drewermann) (empir)   inkonsistent mit
p(kirche,drewermann) (3)
n(kirche,pille) (empir)   inkonsistent mit  p(kirche,pille) (3)
n(papst,drewermann) (empir)   inkonsistent mit  p(papst,drewermann)
(3)
n(papst,pille) (empir)   inkonsistent mit  p(papst,pille) (3)
n(drewermann,pille) (empir)   inkonsistent mit  p(drewermann,pille)
(1)
o(zoelibat,pille) (empir)   inkonsistent mit  n(zoelibat,pille) (2)
o(zoelibat,pille) (empir)   inkonsistent mit  p(zoelibat,pille) (3)
p(drewermann,pille) (1)   inkonsistent mit  n(drewermann,pille)
(empir)
n(ego,pille) (2)   inkonsistent mit  p(ego,pille) (empir)
n(ego,drewermann) (2)   inkonsistent mit  p(ego,drewermann) (empir)
n(zoelibat,pille) (2)   inkonsistent mit  o(zoelibat,pille) (empir)
n(zoelibat,pille) (2)   inkonsistent mit  p(zoelibat,pille) (3)
p(kirche,pille) (3)   inkonsistent mit  n(kirche,pille) (empir)
p(kirche,drewermann) (3)   inkonsistent mit  n(kirche,drewermann)
(empir)
p(papst,pille) (3)   inkonsistent mit  n(papst,pille) (empir)
p(papst,drewermann) (3)   inkonsistent mit  n(papst,drewermann)
(empir)
p(zoelibat,pille) (3)   inkonsistent mit  o(zoelibat,pille) (empir)
p(zoelibat,pille) (3)   inkonsistent mit  n(zoelibat,pille) (2)
yes

?- generiere_matrix.

yes

?- matrix.

1  [p n n p n p]
2  [n p p n p n]
3  [n p p n p n]
4  [p n n p n n]
5  [n p p n p o]
6  [p n n n o p]
yes
```

```
?- verbessere_matrix.

Alte Matrix S
1  [p n n p n p]
2  [n p p n p n]
3  [n p p n p n]
4  [p n n p n n]
5  [n p p n p o]
6  [p n n n o p]

Differenz N-P: 3 in Zeile/Spalte: 4

Neue Matrix S*:
1  [p,n,n,n,n,p]
2  [n,p,p,p,p,n]
3  [n,p,p,p,p,n]
4  [n,p,p,p,p,p]
5  [n,p,p,p,p,o]
6  [p,n,n,p,o,p]
yes

?- verbessere_matrix.

Differenz N-P: 3 in Zeile/Spalte: 1

Neue Matrix S*:
1  [p,p,p,p,p,n]
2  [p,p,p,p,p,n]
3  [p,p,p,p,p,n]
4  [p,p,p,p,p,p]
5  [p,p,p,p,p,o]
6  [n,n,n,p,o,p]
yes

?- verbessere_matrix.

Differenz N-P: 2 in Zeile/Spalte: 6

Neue Matrix S*:
1  [p,p,p,p,p,p]
2  [p,p,p,p,p,p]
3  [p,p,p,p,p,p]
4  [p,p,p,p,p,n]
5  [p,p,p,p,p,o]
6  [p,p,p,n,o,p]
yes

?- verbessere_matrix.
no

?- objekte.

ego kirche papst drewermann zoelibat pille
```

Die Matrix kann nach dem dritten Schritt nicht weiter verbessert werden. Zur Herstellung von Balance muß somit die Relation zwischen dem vierten und sechsten

Element geändert werden, also zwischen `drewermann-pille`. Da die Originalrelation negativ war, ist die Beziehung in P oder O umzuwandeln. Diese Modell-Lösung ist intuitiv einsichtig und wurde von der Versuchsperson "akzeptiert". Verändert man diese Relationen entsprechend in der Wissensbasis, läßt sich keine Ambivalenz-Relation ableiten.

An dem konkreten Beispiel läßt sich noch einmal kurz der Modelltyp, die strukturalistische Deutung von Computermodellen und der Vorteil des wissensbasierten Ansatzes erläutern.

Eingeordnet in das Klassifikationsschema von Abelson (Abb. 2.9) liegt mit dem Programm die Simulation eines kognitiven Prozesses vor. In dem Schema von Troitzsch (Tab. 2.1) entspräche das Programm einem qualitativen, deterministischen, diskreten und dynamischen Mikromodell.

Das Verhältnis von Daten und Programm einerseits, sowie Modell und intendierter Anwendung andererseits, kann für die vorliegende Theorie aus strukturalistischer Sicht wie folgt interpretiert werden. Die für einen Programmlauf eingegebenen Daten einer Versuchsperson können als intendierte Anwendung der Theorie betrachtet werden. Das Programm entspricht den Axiomen, welche die Menge der Modelle festlegen. Der Verbesserungs-Algorithmus erzeugt aus den unbalancierten Input-Daten eine Folge von Verbesserungen, so daß die so veränderten Daten das Fundamentalgesetz (3) von Definition 8 erfüllen und zusammen mit den Eingangsdaten ein Modell der Theorie sind. Da jede Verbesserung S' zu S implizit "auf der Zeitachse" abläuft, ist auch die Voraussetzung $t' > t$ erfüllt. Entsprechen die vom Computer erzeugten Verbesserungen beobachteten Verbesserungen von Versuchspersonen, so gehören diese ebenfalls zur intendierten Anwendung.

Die wichtigsten Vorzüge des wissensbasierten Ansatzes im Vergleich zur konventionellen Modellierung lassen sich an der konkreten Theorie nochmals unmittelbar aufzeigen.

- In dem Modell ist das theoretische Wissen explizit und zusammenhängend codiert. Es ist nicht verstreut und implizit im Programmcode versteckt. Der theoretische Bereich ist vollständig vom nicht-theoretischen Programmteil isoliert. Die Regeln bilden ein Modul, und auch der Verbesserungs-Algorithmus ist sauber getrennt von anderen Teilen.
- Es macht keine Probleme, ohne Änderung des Programmcodes, z.B. für Experimentierzwecke, Regeln zu entfernen, auszutauschen oder neue hinzuzufügen.
- Die Definitionen können ohne große Änderung in PROLOG natürlich und leicht lesbar umgesetzt werden.
- Die Schlußfolgerungen für den nicht-algorithmischen Teil lassen sich rechtfertigen.

Während der deklarative Teil ohne weiteres modifiziert werden kann, trifft dies auf die algorithmischen Komponente nicht zu. Der oben von Irle (1975) angesprochene Vorschlag, den Verbesserungs-Algorithmus in Einklang mit empirischen Ergebnissen um Zusatzhypothesen zu erweitern, kann nicht ohne massiven Eingriff in diesen Programmbereich erfolgen.

Das Computermodell könnte noch etwas benutzerfreundlicher ausgebaut werden

und dann dazu verwendet werden, um etwas systematischer zu testen, ob die Theorie "plausible" Schlußfolgerungen generiert. Erste Plausibilitätstests könnten mit subjektiven Einstellungsräumen des Modellbauers durchgeführt werden. In einer stärkeren Variante - die bereits in dem Beispiel oben benutzt wurde - könnten Versuchspersonen mit der Programmlösung konfrontiert, anhand der Schlußrechtfertigungskomponente auf ihre Widersprüche hingewiesen und nach Akzeptanz der Lösung gefragt werden. In einer noch stärkeren Variante könnten Modellprotokolle mit Protokollen von Versuchspersonen auf I/O-Äquivalenz verglichen werden. Auf die Testproblematik können wir im Rahmen dieser Arbeit aber nicht eingehen.

8. Makrostrukturelle Konsequenzen von Balance

Die Wurzeln der Balancetheorien liegen in der Psychologie, und die bislang betrachteten Theorien hatten intendierte Anwendungen, die eher im psychologischen Bereich lagen. In der Folge entwickelte sich nun ein balancetheoretischer Theorienstrang, der über rein psychologische Anwendungen hinausging und auf die Analyse sozialer Netze ausgedehnt wurde. Ausgangspunkt für diese theoretische Entwicklung ist das Modell der "strukturellen Balance" von Cartwright/Harary. In der Literatur gilt diese Arbeit aus dem Jahr 1956 als eigentliche Verallgemeinerung und Präzisierung der Theorie von Heider und als Begründung der balancetheoretischen Analyse sozialer Netzwerke.

8.1 Aspekte der Graphentheorie

Cartwright/Harary präzisieren und verallgemeinern die Theorie von Heider graphentheoretisch, und auch die folgenden Modelle bauen auf graphentheoretischen Konzepten auf. Wir müssen deshalb im folgenden kurz einige ausgewählte Aspekte der Graphentheorie betrachten.

8.1.1 Graphen und Graphtypen

Wir haben mit der Definition des Heider- und AR-Graphen den Begriff des "Graphen" bereits verwendet ohne ihn explizit eingeführt zu haben und holen dies hier nach.[56] Ein *Graph* G ist definiert durch ein geordnetes Paar $G := \langle X,R \rangle$ mit einer endlichen[57], nicht-leeren Menge X und einer Menge $R = \{\{x,y\} | x,y \in X\}$. Die Elemente von X heißen *Knoten* ("node") oder *Punkte* ("point"), die Elemente von R *Kanten* ("edge"). Jede Kante $r = \{x,y\} \in R$ verbindet x mit y, man sagt: x ist mit y benachbart oder adjazent. Beispiele für Graphen sind:
$G1 = \langle \{a,b,c,d\}, \{\{a,b\},\{a,c\},\{b,c\}\} \rangle$
$G2 = \langle \{a\}, \{\} \rangle$.
Graphen lassen sich geometrisch darstellen. Den Knoten entsprechen hierbei Punkte, den Kanten Kurven oder Linien, welche die Punkte verbinden. Die folgenden Diagramme repräsentieren die beiden Graphen G1 und G2.

---

[56] Einführungen in die Graphentheorie sind z.B. Harary (1969), Jungnickel (1990) oder Tutte (1984).

[57] In der Graphentheorie werden auch unendliche Graphen behandelt, die aber in unserem Kontext nicht relevant sind.

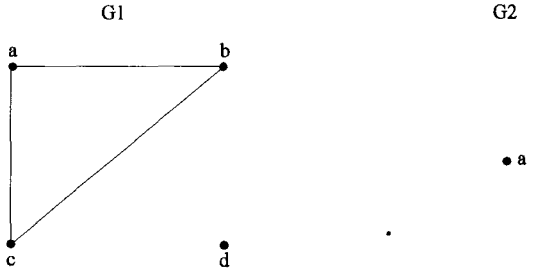


Abb. 8.1: Geometrische Darstellung der Graphen G1 = ⟨{a,b,c,d},{{a,b},{a,c},{b,c}}⟩ und G2 = ⟨{a},{}⟩

Den Objekten in Heider-Graphen entsprechen in graphentheoretischer Terminologie Knoten, den Relationen Kanten, und die Strukturen in Abb. 6.2 (S.120) können als geometrische Darstellung von Heider-Graphen interpretiert werden. Allerdings haben wir in Heider-Graphen zwei Typen von Relationen, die wir mit dem bislang eingeführten Instrumentarium noch nicht unterscheiden können. Angewendet auf soziale Netze können Knoten Personen - oder allgemeiner: Akteure (z.B. auch Organisationen, Staaten etc.) - repräsentieren und Kanten soziale Relationen, wie Freundschafts-, Austausch- oder Kommunikationsbeziehungen.

In unserem Zusammenhang sind bestimmte Varianten oder Typen von Graphen relevant. Ein *gerichteter* Graph oder *Digraph* ("Directed Graph") ist ein Paar $G = \langle X,R \rangle$, bestehend aus einer endlichen, nicht-leeren Menge X und einer Menge $R \subseteq X \times X$ von *geordneten Paaren* $\langle x,y \rangle$ mit $x,y \in X$ $(x \neq y)$. Jedes geordnete Paar $\langle x,y \rangle \in R$ ist eine Kante des Digraphen. Der einzige Unterschied zwischen der Definition eines Graphen und der eines Digraphen ist, daß die Kanten eines Graphen zweielementige *Mengen* von Knoten sind, also ungeordnete Paare, während die Kanten eines Digraphen *geordnete Paare* von Knoten sind. Im Falle eines Digraphen spricht man deshalb auch von gerichteten Kanten und stellt diese im Diagramm als Pfeile dar. Der gerichtete Graph
G3= ⟨{a,b,c}{⟨a,b⟩,⟨a,c⟩,⟨c,a⟩,⟨c,b⟩}⟩
könnte danach in einem Diagramm wie in Abb. 8.2 gezeichnet werden.

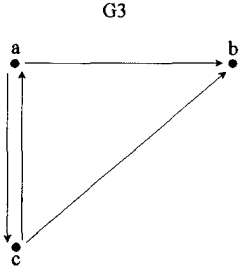


Abb. 8.2: Der gerichtete Graph G3= ⟨{a,b,c},{⟨a,b⟩,⟨a,c⟩,⟨c,a⟩,⟨c,b⟩}⟩

Dieser gerichtete Graph könnte z.B. drei Personen a, b und c repräsentieren, wobei a und c jeweils Informationen an b geben (aber nicht umgekehrt) und a und c Informationen beidseitig austauschen.

Einen *bewerteten* Graphen oder *S-Graphen* ("Signed Graph") erhält man aus einem Graphen, wenn man eine Teilmenge von Kanten als positiv und die verbleibende Teilmenge als negativ betrachtet (Harary 1969). Damit läßt sich z.B. eine Relation und deren Gegenteil darstellen. Ordnet man positiven Kanten +1, negativen Kanten -1 zu, dann läßt sich ein bewerteter Graph mit Knoten X und Kanten R wie folgt schreiben:
$G = \langle X,\{\langle\{x,y\},z\rangle |\ x,y \in X \text{ und } z \in \{+1,-1\}\}\rangle$.
Die Heider-Graphen von S.120 können dann als bewertete S-Graphen aufgefaßt werden, wobei die durchgezogenen Linien positive, gestrichelte Linien negative Kanten symbolisieren.

Die Kombination von Digraphen und S-Graphen ergibt *S-Digraphen*. In einem S-Digraphen gibt es ausschließlich positive und negative gerichtetete Kanten. S-Digraphen können wie S-Graphen repräsentiert werden mit dem Unterschied, daß die Menge $\{x,y\}$ durch das geordnete Paar $\langle x,y\rangle$ ersetzt wird.

Schließlich kann mit Graphen nicht nur *ein* Relationstyp und sein Gegenteil dargestellt werden, sondern *beliebig viele* Relationstypen. Ein *Graph vom Typ* τ, $\tau = 1,2,3$ ... liegt vor, wenn τ verschiedene Relationen auf der Menge X definiert sind (Cartwright/Harary 1956). In einem Graphen vom Typ 2 gibt es also zwei verschiedene Relationen (z.B. Macht- und Kommunikationsbeziehungen) und in einem Graphen vom Typ 5 entsprechend fünf unterschiedliche Relationen. Grafisch könnte man die diversen Relationstypen mit bestimmten Farben repräsentieren.

Kombinieren wir alle bisher eingeführten graphentheoretischen Konzepte zu einem *S-Digraph vom Typ 2*, so existieren in diesem Graphen zwei verschiedene (gerichtete) Relationen, von denen jede positiv oder negativ sein kann. Dieser Graph könnte geometrisch mit vier verschiedenen Kantenarten dargestellt werden, z.B. blaue durchgezogene (positive erste Relation), blaue gestrichelte (negative erste Relation), rote durchgezogene (positive zweite Relation) und rote gestrichelte (negative zweite Relation) Linien.

Schließlich benötigen wir noch den Begriff des Sub-Graphen und der Vollständigkeit. Ein Graph $G' = \langle X',R'\rangle$ ist *Sub-Graph* eines Graphen $G = \langle X,R\rangle$ gdw $X' \subseteq X$ und $R' \subseteq R$. Beispielsweise ist $\langle\{a,c\}, \{\langle a,c\rangle, \langle c,a\rangle\}\rangle$ Sub-Graph von G3 und G2 ist Sub-Graph von G1. In einem *vollständigen Graphen* sind alle Knotenpaare miteinander verbunden. G3 ist beispielsweise vollständig, ebenso die Heider-Graphen von Abb. 6.2, was mit Axiom (7) von Definition 1 (HT) auch explizit gefordert wird. Hingegen ist G1 unvollständig.

8.1.2 Verknüpfungen in Graphen

Eine der elementaren Eigenschaften eines jeden Graphen ist die Verknüpfung (Harary 1969: 13, Jungnickel 1990: 17). Es sei $\langle r_1,...,r_n\rangle$ eine Folge von Kanten eines ungerichteten Graphen G. Wenn es Knoten $x_0,...,x_n$ gibt mit $r_i=\{x_{i-1},x_i\}$ für i=1,...,n, so heißt die Folge ein *Kantenzug* K ("walk"). Dies läßt sich anschaulicher wie folgt schreiben:

$$K: x_0 \overset{r_1}{\text{——}} x_1 \overset{r_2}{\text{——}} x_2 \text{——} \ldots \overset{r_{n-1}}{\text{——}} x_{n-1} \overset{r_n}{\text{——}} x_n.$$

x_0 bzw. x_n heißen der *Anfangs-* bzw. *Endpunkt* des Kantenzugs K. Im Fall $x_0=x_n$ spricht man von einem *geschlossenen Kantenzug* ("closed walk"). Als Beispiel betrachten wir den ungerichteten Graphen in Abb. 8.3.

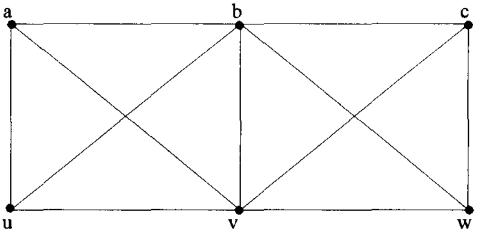


Abb. 8.3: Beispiele für Kantenzüge in einem ungerichteten Graphen (nach Jungnickel 1990: 18)

In Abb. 8.3 ist ⟨{a,b},{b,c},{c,v},{v,b},{b,c}⟩ ein (nicht geschlossener) Kantenzug. Einfachheitshalber geben wir Kantenzüge im folgenden durch Knotenfolgen wieder und schreiben für den eben genannten Kantenzug: ⟨a,b,c,v,b,c⟩. a wäre hierbei der Anfangspunkt, c der Endpunkt. Wenn die Kanten r_i paarweise verschieden sind, liegt ein *Weg* vor ("trail"), im geschlossenen Fall ein *Kreis* ("closed trail"). Beispielsweise wäre in Abb. 8.3 die Folge ⟨a,b,c,v,b,u⟩ ein Weg und ⟨a,b,c,v,b,u,a⟩ ein Kreis. Falls auch die Knoten x_i paarweise verschieden sind, handelt es sich um einen *Pfad* ("path", einfacher Weg). Danach wäre <a,b,c,v,u> ein Pfad. Ein Kreis, für den - abgesehen von $x_0=x_n$ - die x_j paarweise verschieden sind, ist ein *Zyklus* ("cycle", einfacher Kreis). In Abb. 8.3 ist ⟨a,b,c,w,v,u,a⟩ ein möglicher Zyklus. Ist x_n der Endpunkt von K, dann ist n die *Länge* eines Kantenzuges (Weges, Kreises oder Zyklus), und ein *n-Zyklus* ist ein Zyklus der Länge n. Danach hat der Kantenzug ⟨a,b,c,v,b,c⟩ die Länge 5, und ⟨a,b,c,w,v,u,a⟩ wäre ein 6-Zyklus.

Die eben besprochenen Begriffe lassen sich analog auf Digraphen anwenden. Es sei $\langle r_1,...,r_n\rangle$ eine Folge von Kanten eines Digraphen G. Wenn es Knoten $x_0,...,x_n$ gibt mit $r_i=\langle x_{i-1},x_i\rangle$ für i=1,...,n, heißt die Folge ein Kantenzug ("walk") in einem Digraphen. Die anderen Begriffe werden analog abgewandelt.

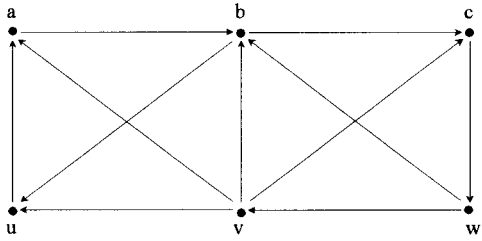


Abb. 8.4: Beispiele für Kantenzüge in einem gerichteten Graphen

In Abb.8.4 ist beispielsweise ⟨a,b,c,w,b,u⟩ ein Weg, aber kein Pfad, ⟨a,b,c,w,v⟩ ist ein Pfad, ⟨a,b,c,w,b,u,a⟩ ein Kreis, aber kein Zyklus und ⟨a,b,c,w,v,u,a⟩ ein Zyklus.

Eine Variation des Zyklus-Begriffs in Digraphen ist der Semi-Zyklus. Ein *Semi-Zyklus* ist definiert wie ein Zyklus mit der Abschwächung, daß die Elemente der Folgenmenge K genau eine Kante $r_i = \langle x_{i-1},x_i\rangle$ oder $\langle x_i, x_{i-1}\rangle$ enthalten dürfen, also: $\langle\langle x_0,x_1\rangle$ oder $\langle x_1,x_0\rangle$, $\langle x_1,x_2\rangle$ oder $\langle x_2,x_1\rangle$, ... , $\langle x_n,x_0\rangle$ oder $\langle x_0,x_n\rangle\rangle$.
In einem Semi-Zyklus ist somit die Richtung der Pfeile irrelevant.

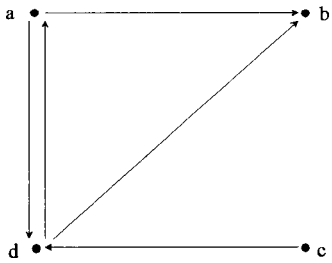


Abb. 8.5: Ein gerichteter Graph mit drei Semi-Zyklen (nach Cartwright/Harary 1956: 286)

In Abb. 8.5 gibt es beispielsweise drei Semi-Zyklen:
K1=⟨⟨a,d⟩, ⟨d,a⟩⟩
K2=⟨⟨a,d⟩, ⟨d,b⟩, ⟨a,b⟩⟩
K3=⟨⟨d,a⟩, ⟨a,b⟩, ⟨d,b⟩⟩.
Der erste Semi-Zyklus ist auch ein Zyklus, die letzten beiden Semi-Zyklen sind keine Zyklen.

In einem S-Graphen kann jeder Zyklus und Semi-Zyklus *bewertet* werden in Abhängigkeit von der Bewertung der Kanten. Wenn man wie oben festlegt, daß der Wert einer Kante +1 bzw. -1 ist, wenn die Kante positiv bzw. negativ ist, dann ergibt sich der Wert eines Zyklus bzw. Semi-Zyklus als Produkt der Werte der Kanten. Allgemein gilt: Ein (Semi-)Zyklus ist positiv, wenn er eine gerade Anzahl von negativen Kanten enthält, sonst negativ.

Relationstheoretisch korrespondieren Graphen ⟨X,R⟩ Relationen R auf X, die im Fall von ungerichteten Graphen symmetrisch sind. Wir betrachten im folgenden die relationale und graphentheoretische Interpretation von R als austauschbar und verwenden beide Darstellungsarten.

8.2 Strukturelle Balance

Mit dem eben eingeführten Instrumentarium läßt sich das Grundmodell von Heider - wie bereits angedeutet - graphentheoretisch reformulieren und in einem nächsten Schritt verallgemeinern. Diese graphentheoretische Verallgemeinerung wurde erstmals von Cartwright/Harary (1956) durchgeführt und betrifft folgende Punkte:

1. Es werden symmetrische und nicht-symmetrische Relationen zugelassen;
2. Die Kardinalität n der Objekte wird erweitert auf $n > 3$;
3. Relationen unterschiedlichen Typs sind erlaubt;
4. Die Anwendung wird ausgedehnt auf "objektiv" beobachtbare soziale und andere Relationen.

Die Erweiterung von der kognitiven Sichtweise auf "objektiv" beobachtbare Sozialrelationen macht das Modell zu einer Strukturtheorie interpersoneller Beziehungen und damit für Soziologen eher zugänglich und interessanter.[58] In Absetzung zur Heider-Theorie bezeichnen die Autoren ihr Modell deshalb als "strukturelle Balance". Versteht man unter Mikro-Struktur die Struktur des Beziehungsnetzes auf individueller Ebene und unter Makro-Struktur die Struktur des Netzes als Ganzem, so untersucht das Modell nun die logischen Zusammenhänge zwischen Mikro- und Makro-Struktur.

8.2.1 Polarisierung und Einmütigkeit

Mit den graphentheoretischen Begriffen lassen sich die Heider-Triaden neu beschreiben als (gerichtete oder ungerichtete) S-Graphen vom Typ 1. Den Elementen der Objektmenge O entsprechen Knoten, den Relationen P und N bewertete Kanten. Wenn wir die Relationen als *symmetrisch* auffassen, sind die Heider-Triaden als *ungerichtete* bewertete Kanten zu interpretieren, und der balancierten Triade (b) in Abb. 6.2 entspräche dann der Graph $G = \langle\{p,o,x\},\{\langle\{p,o\},+1\rangle,\langle\{p,x\},-1\rangle,\langle\{o,x\},-1\rangle\}\rangle$.

Jeder der S-Graphen in Abb. 6.2 enthält genau einen Zyklus $\langle p,o,x\rangle$. Der Wert der Zyklen (a) bis (d) ist +1, der Wert der Zyklen (e) bis (h) ist -1. (a) bis (d) sind genau die balancierten, die anderen - mit Ausnahme von (h) - die unbalancierten Triaden. Alle balancierten Triaden haben also positive, unbalancierte negative Zyklen. Cartwright/Harary benutzen nun dieses Charakteristikum als allgemeines Kriterium für Balance *mit beliebig großer Knotenmenge* und definieren Gleichgewicht in Abhängigkeit vom Wert der Zyklen.

Definition 1

Ein *S-Graph* (beliebiger Kardinalität) ist balanciert genau dann, wenn alle Zyklen positiv sind.

---

[58] Hummell/Sodeur (1987) weisen darauf hin, daß damit die nicht unproblematische Annahme der Korrektheit der Wahrnehmung seitens der Akteure getroffen werden muß, d.h. die kognitiven Repräsentationen der Relationen dürfen nicht von den "tatsächlichen" Relationen abweichen. Analog formuliert Opp (1984), daß Personen die Konfiguration zumindest in der gleichen Art identifizieren müssen und die "Hypothese der kongruenten Wahrnehmung" gelten muß.

Man beachte, daß die von Heider als indefinit eingestufte Triade (h) von Abb. 6.2 (S.120) nach dieser Definition unbalanciert ist.

Abb. 8.6 illustriert die Definition anhand von vier S-Graphen mit je vier Knoten. Jeder dieser Graphen enthält sieben Zyklen: ⟨a,b,c,a⟩, ⟨a,b,d,a⟩, ⟨b,c,d,b⟩, ⟨a,c,d,a⟩, ⟨a,b,c,d,a⟩, ⟨a,b,d,c,a⟩, ⟨b,c,a,d,b⟩.

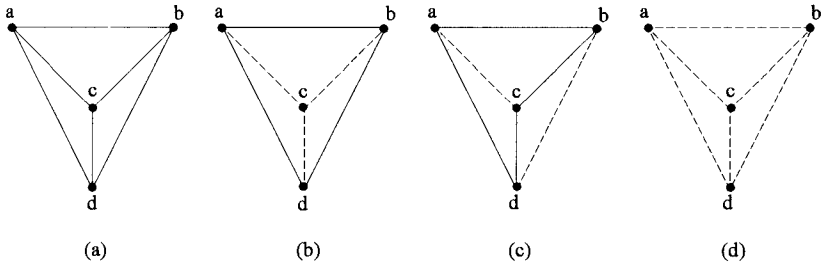


Abb. 8.6: Vier S-Graphen mit unterschiedlich bewerteten Kanten (nach Cartwright/Harary 1956). Durchgezogene Linien symbolisieren mit +1, gestrichelte Linien mit -1 bewertete Kanten

Nach Definition 1 sind die S-Graphen (a) und (b) balanciert, da alle Zyklen ein positives Produkt haben, die S-Graphen (c) und (d) sind unbalanciert, da es Zyklen mit negativem Wert gibt, wie z.B. ⟨a,b,c,a⟩.

Die Ausweitung der Balancedefinition auf *gerichtete* S-Digraphen mit beliebig vielen Knoten erfolgt analog der obengenannten Definition mit Semi-Zyklen.

Definition 2

Ein *S-Digraph* ist balanciert genau dann, wenn alle Semi-Zyklen positiv sind.

Abb. 8.7 zeigt drei Graphen, die als Heider-Triaden mit *gerichteten* Beziehungen interpretierbar sind. Mit S-Digraphen kann ausgedrückt werden, daß von p *und* o Relationen ausgehen können, während von der Nicht-Person x keine Relationen ausgehen.

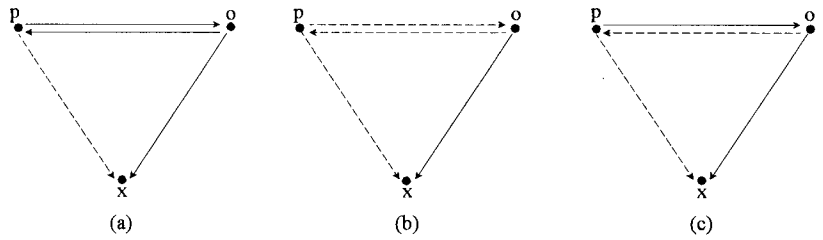


Abb. 8.7: Drei S-Digraphen mit unterschiedlich bewerteten (+1, -1) gerichteten Kanten (nach Cartwright/Harary 1956)

Jede dieser Triaden enthält drei Semi-Zyklen: ⟨⟨p,o⟩,⟨o,p⟩⟩ und ⟨⟨p,o⟩,⟨o,x⟩,⟨p,x⟩⟩ und ⟨⟨o,p⟩,⟨p,x⟩,⟨o,x⟩⟩.

In Struktur (a) sind die Semi-Zyklen ⟨⟨p,o⟩,⟨o,x⟩,⟨p,x⟩⟩ und ⟨⟨o,p⟩,⟨p,x⟩,⟨o,x⟩⟩ negativ, in Struktur (c) ist der Semi-Zyklus ⟨⟨p,o⟩,⟨o,p⟩⟩ negativ. Beide Triaden sind folglich unbalanciert. Nur Triade (b) enthält keine negativen Semi-Zyklen und ist damit balanciert. Diese Ergebnisse können in konsistenter Weise mit der Heider-Theorie verknüpft werden: Wenn zwischen zwei Personen eine beidseitig positive Relation vorliegt, dann ist die Triade nur balanciert, wenn beide Personen zum Objekt x eine beidseitig positive oder beidseitig negative Relation haben (d.h. (a) ist unbalanciert). Wenn zwischen zwei Personen eine beidseitig negative Relation vorliegt, dann ist die Triade nur balanciert, wenn eine Person zu x eine positive, die andere zu x eine negative Relation hat (d.h. (b) ist balanciert). Liegen zwischen beiden Personen unterschiedliche Relationen vor, so ist die Triade in jedem Fall unbalanciert (d.h. (c) ist unbalanciert).

Berücksichtigt man Heiders Unterscheidung in L- und U-Relationen, so läge ein S-Graph vom Typ 2 vor. Wir hätten dann also zwei Relationen - L und U - die jeweils mit +1 und -1 bewertbar sind. Das einfachste Verfahren wäre, die Linientypen zu ignorieren, was wohl auch der Intention Heiders am nächsten kommt.

Definition 3
Ein S-Graph vom Typ 2 ist balanciert genau dann, wenn alle Zyklen positiv sind.

Statt der oben eingeführten Schreibweise für positive und negative Kanten von S-Graphen verwenden wir im folgenden die besser lesbare relationale Schreibweise, die analog auch für Digraphen gelten soll.

Definition 4
Wenn G=⟨X,R⟩ ein S-Graph ist dann gilt:
Für alle x,y ∈ X: xPy gdw ⟨{x,y},+1⟩
Für alle x,y ∈ X: xNy gdw ⟨{x,y},-1⟩

Wir kommen damit zur theoretisch bedeutsamsten Folgerung dieses Modells, dem Strukturtheorem. Es behauptet einen logischen Zusammenhang zwischen balancierten Mikro-Strukturen und Eigenschaften der Makro-Struktur.

(Struktur-)Theorem (Cartwright/Harary 1956)
Sei G = ⟨X,R⟩ ein S-Graph. Dann ist G balanciert gdw es U,V ⊆ X gibt, so daß gilt:
(1) X = U ∪ V
(2) U ∩ V = ∅
(3) Für alle x,y ∈ X: wenn xPy dann x,y ∈ U oder x,y ∈ V
(4) Für alle x,y ∈ X: wenn xNy dann (x ∈ U und y ∈ V) oder (x ∈ V und y ∈ U)

Der Beweis des Theorems findet sich in Harary (1953). Das Strukturtheorem besagt informell, daß ein bewerteter Graph oder S-Graph balanciert ist genau dann, wenn sich seine Knoten in zwei disjunkte Teilmengen zerlegen lassen, so daß jede positive Kante zwei Knoten der gleichen Teilmenge und jede negative Kante zwei Knoten aus

verschiedenen Teilmengen verbindet. Eine so zerlegte Knotenmenge bildet damit eine Partition. Unter Anwendung auf soziale Gruppen bedeutet das Strukturtheorem, daß eine balancierte Gruppe *notwendigerweise* in zwei disjunkte Teilgruppen zerfällt mit *positiven* Relationen *innerhalb* der Teilgruppen und *negativen* Relationen *zwischen* den Teilgruppen. Mit anderen Worten: unter Balance findet eine *Polarisierung* des Gesamtsystems in zwei antagonistische Sub-Gruppen statt, die man als *Cliquen* bezeichnen kann. Ist die Menge N leer, d.h. gibt es keine negativen Relationen, so existiert nur *eine* Gruppe, und man spricht von *Einmütigkeit*.

Betrachten wir einige Beispiele für balancierte Strukturen und die Rolle, die das Strukturtheorem dabei spielen kann. Unter Verwendung des Strukturtheorems sieht man sofort, daß der Graph in Abb. 8.8 balanciert ist mit den beiden Teilmengen U={a,b,c,d} und V={e,f,g,h}.

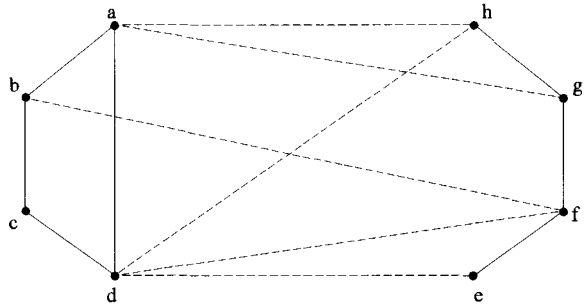


Abb. 8.8: Ein balancierter S-Graph mit den beiden Teilmengen U={a,b,c,d} und V={e,f,g,h} (nach Cartwright/Harary 1956)

Bei dem Graphen in Abb. 8.8 handelt es sich um einen unvollständigen Graphen, bei dem nicht *alle* Knotenpaare durch P- oder N-Kanten verbunden sind. Nicht vorhandene Kanten können in diesem Fall als Indifferenz gedeutet werden. Die Graphen von Abb. 8.6 (S.171) sind hingegen alle vollständig. In diesen Graphen sind die Cliquenmengen ebenfalls leicht zu ersehen. Graph (b) in Abb. 8.6 z.B. zerfällt in die beiden Teilmengen U={a,b,d} und V={c}). In Graph (a) von Abb. 8.6 ist mit U={a,b,c,d} und V=∅ eine Teilmenge leer, da es ausschließlich P-Relationen gibt. In diesem Fall gibt es also nur *eine* Gruppe (Einmütigkeit).

In den unvollständigen Graphen von Abb. 8.9 und 8.10 ist die Balance weniger offensichtlich. In diesem Fall ist das Strukturtheorem sehr nützlich, da die beiden Teilmengen relativ leicht bestimmt werden können und damit der Gleichgewichtszustand feststeht.

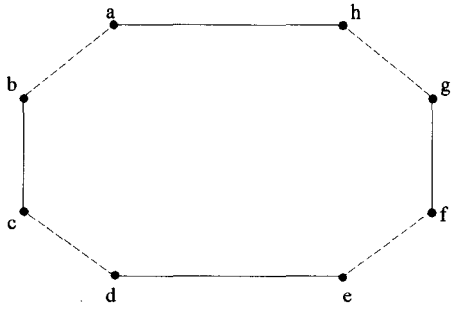


Abb. 8.9: Balancierter S-Graph mit den Teilmengen U={a,h,d,e} und V={b,c,g,f} (nach Cartwright/Harary 1956)

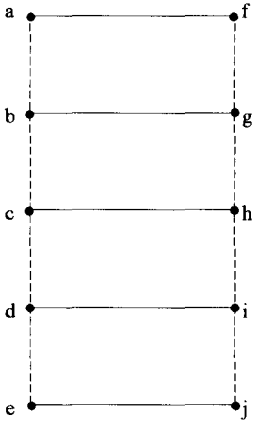


Abb. 8.10: Balancierter S-Graph mit den Teilmengen U={a,f,c,h,e,j} und V={b,g,d,i} (nach Cartwright/Harary 1956)

Allgemein gilt, daß in *vollständigen* balancierten Graphen die Zerlegung in zwei Teilgruppen immer eindeutig ist, in *unvollständigen* balancierten Graphen ist eine Zerlegung *nicht immer eindeutig* (Hummell/Sodeur 1984: 534-535).

Das Strukturtheorem hat einige wichtige inhaltliche Konsequenzen (vgl. Cartwright/Harary 1956: 287). Angenommen, zwei durch positive Kanten verknüpfte Personen können sich gegenseitig positiv beeinflussen, und zwei durch negative Kanten verknüpfte Personen können sich nur negativ beeinflussen. Ein S-Graph, der die Sympathie- und Antipathie-Relationen einer Gruppe widerspiegelt, gibt dann auch die Einflußstruktur in dieser Gruppe wider. Wenn in Abb. 8.10 a versucht, b dahin zu bringen, etwas anzuerkennen, wird b mit Mißbilligung reagieren. Wenn b hingegen versucht, g zur Mißbilligung zu bewegen, wird er erfolgreich sein. Also ist der (indirekte) Einfluß von a auf g negativ. Eine Konsequenz des Strukturtheorems ist mithin, daß in balancierten Gruppen innerhalb einer Clique der Einfluß von Person zu Person positiv sein muß - ja sogar wenn er Individuen außerhalb der Clique betrifft - und der Einfluß muß negativ sein, wenn er von einer Person von Clique A zu einer

Person von Clique B geht. Unter der Voraussetzung der Balance wird deshalb jeder Meinungseinfluß innerhalb von Cliquen zu Homogenität und zwischen den Cliquen zu gegensätzlichen Meinungen tendieren.

8.2.2 Balanceprinzip und intendierte Anwendungen

Wie lautet nun das Balanceprinzip in dem Modell von Cartwright/Harary, und worin bestehen die intendierten Anwendungen? Während bei Heider eine Triade entweder balanciert oder unbalanciert ist, tritt bei einer größeren Menge von Elementen Balance normalerweise nur in einem mehr oder weniger hohen Grad auf. Die bislang betrachteten vollständig balancierten S-Graphen sind unter empirischer Sichtweise eher der Ausnahmefall. Cartwright/Harary definieren als Maß für die Balance eines Graphen G deshalb einen Gleichgewichtsgrad über die Anzahl positiver Zyklen, bezogen auf die Anzahl aller möglichen Zyklen.

Definition 5
Sei G ein S-Graph, c(G) := Anzahl aller Zyklen in G und c+(G) := Anzahl der positiven Zyklen in G. Dann ist

$$b(G) := \frac{c+(G)}{c(G)} \quad \text{mit } 0 \leq b(G) \leq 1$$

der Gleichgewichtsgrad von G.

Der Balancegrad schwankt zwischen 0 und 1. Je näher der Index an 1 liegt, um so balancierter ist der Graph, und je näher er an 0 liegt, um so unbalancierter ist er. Vollständig balancierte Graphen (wie die Graphen in Abb. 8.8, 8.9 oder 8.10) haben den Gleichgewichtsgrad 1. Der Gleichgewichtsgrad für S-Digraphen kann analog über Semi-Zyklen definiert werden.

Das zentrale Axiom der Heider-Theorie war, daß unbalancierte Triaden über die Zeit in balancierte überführt werden. Entsprechend heißt das zentrale Axiom der Cartwright-Harary-Theorie, *daß der Gleichgewichtsgrad von S-Graphen über die Zeit gleich bleibt oder größer wird.* Angewendet auf das Strukturtheorem bedeutet dies, daß mit der Zeit und der *Tendenz zu Gleichgewicht* auch eine *Tendenz zur Polarisierung* einer Gruppe in zwei Sub-Gruppen beobachtbar ist.

Eine mengentheoretische axiomatische Darstellung der Theorie mit genau dem eben genannten fundamentalen Gesetz findet sich bei Sukale (1971).[59] Eine strukturalistische Rekonstruktion liefert Stephan (1990). Stephan formuliert das inhaltliche Axiom aber strenger als Sukale. Er fordert, daß unbalancierte Systeme den Zustand des Ungleichgewichts verlassen *müssen*, d.h. im Unterschied zur Rekonstruktion von Sukale darf der Balancegrad über ein hinreichend langes Zeitintervall nicht gleich bleiben (Stephan 1990: 74). Wir kommen auf diesen Sachverhalt bei der Rekonstruktion des T-Graph-Modells noch einmal zurück (Kap. 9.3, S.191).

---

[59] Die mengentheoretische Präzisierung von Sukale ist keine strukturalistische Rekonstruktion. 1971, zum Zeitpunkt der Veröffentlichung der Arbeit von Sukale, erschien gerade das Buch von Sneed. Der strukturalistische Apparat stand Sukale also noch nicht zur Verfügung.

Bei der Frage nach den intendierten Systemen, auf die ihre Theorie angewendet werden soll, verhalten sich Cartwright/Harary ganz nach den Vorstellungen der strukturalistischen Wissenschaftstheoretiker. Sie betrachten die Anwendungen getrennt vom Theoriekern und nennen gleich mehrere intendierte Systeme. Zunächst erwähnen sie, daß die Balancedefinition dazu dienen könnte, kognitive Einheiten, soziale Systeme oder *irgendeine* Konfiguration, welche sowohl eine Relation als auch ihr Gegenteil beinhaltet, zu charakterisieren: "Clearly, our definition of balance may be employed whenever the terms 'point' and 'signed line' can be meaningfully coordinated to empirical data of any sort" (Carwright/Harary 1956: 292). Im Verlauf der Entwicklung ihres Modells beschränken sich die Autoren auf die intendierte Anwendung "soziales System", insbesondere "soziale Gruppen", was sich zum einen in den Beispielen, zum anderen in der verwendeten Terminologie äußert. Beispielsweise wird die Relation P durchgängig als Liking-Relation gedeutet, die Menge X als (soziale) Gruppe bzw. Teilmengen von X als Sub-Gruppe oder Clique. Am Schluß liefern die Autoren eine ganze Liste intendierter Anwendungen: "Although Heider's theory was originally intended to refer only to cognitive structures of an individual person, we propose that the definition of balance may be used generally in describing configurations of many different sorts, such as communication networks, power systems, sociometric structures, systems of orientations, or perhaps neural networks" (Cartwright/Harary 1956: 292). Die ursprüngliche Menge intendierter kognitiver Systeme I(HT) bleibt also bei Cartwright/Harary erhalten, wird aber erheblich expandiert. Das Zitat belegt im übrigen erneut, daß die Anwendungen nicht automatisch fest mit einer Theorie mitgeliefert werden. Wir möchten die ausführlichere Diskussion um die Anwendungsmöglichkeiten der verallgemeinerten Balancetheorien auf Kap. 9.3 verschieben.

Das Cartwright-Harary-Modell wurde im Verlauf der historischen Entwicklung weiter differenziert in das Clustering- und Ranked-Clustering-Modell. Wir gehen noch kurz auf diese Modelle ein, verzichten aber in allen drei Fällen auf eine PROLOG-Implementierung und behalten uns dies für das allgemeinste Modell des transitiven Graphen vor.

8.3 Clustering und Ranked Clustering

Cartwright/Harary haben die Eigenschaften des Gesamtsystems eines Netzes von Personen aufgezeigt, die sich ergeben, wenn für jede einzelne Person das System ihrer Sozialrelationen ausgeglichen ist. Diese "makrostrukturellen" Konsequenzen von Balance bilden in den weiteren Modelldifferenzierungen den Fokus des Interesses.

8.3.1 Erweiterte Gruppierbarkeit

Konsequenz des Strukturtheorems ist, daß in balancierten Strukturen alle Elemente in zwei disjunkte Teilmengen zerlegbar sind, die als Cliquen interpretiert werden können. Falls die interpersonellen Relationen ausgeglichen sind (Mikro-Ebene), gibt es eine

polarisierte Gruppenstruktur (Makro-Ebene), und Tendenz zum Gleichgewicht impliziert Tendenz zur Entwicklung zweier polarisierter Sub-Gruppen. Empirisch werden nun aber Gruppen beobachtet, die in drei, vier und mehr Cliquen zerfallen. Davis (1967) fragt deshalb nach den notwendigen und hinreichenden Bedingungen einer *erweiterten* Gruppierbarkeit (Cluster). Mit "erweiterter Gruppierbarkeit" ist gemeint, daß Graphen in zwei *oder mehr* Teilmengen partitioniert werden können, so daß innerhalb der Teilmengen positive und zwischen den Teilmengen negative Relationen vorliegen.

Die Bedingung für eine *erweiterte* Gruppierbarkeit liegt in einer einfachen Abschwächung des Balancebegriffs unter Betrachtung der problematischen Triade mit drei N-Relationen. Die kritische Triade mit drei negativen Relationen ((h) in Abb. 6.2, S.120) wurde von Heider nicht eindeutig klassifiziert. Nach dem Balancekriterium von Cartwright/Harary ist die Triade unbalanciert, da der Wert des Zyklus negativ ist. Liegt Ausgeglichenheit nach dem Balancekriterium von Cartwright/Harary vor, sprechen Hummell/Sodeur (1984) von *B-Ausgeglichenheit*. Die Triade (h) in Abb. 6.2 wäre also nicht B-ausgeglichen. Triade (h) tritt empirisch aber häufig auf.[60] Davis konnte nun zeigen, daß Gruppen in mehrere Teilgruppen zerfallen, wenn der Balancebegriff dahingehend abgeschwächt wird, daß auch Triade (h) als ausgeglichen betrachtet wird. Hummell/Sodeur (1984) sprechen in diesem Fall von (abgeschwächter) *C-Ausgeglichenheit*. Alle B-ausgeglichenen Triaden wären demnach auch C-ausgeglichen, das umgekehrte gilt nicht. Die Triaden (a) bis (d) in Abb. 6.2 sind danach B- und C-ausgeglichen, die Triaden (e), (f), (g) sind weder B- noch C-ausgeglichen, und die Triade (h) ist C-, aber nicht B-ausgeglichen.

Allgemein gilt:

Eine Triade oder ein Zyklus ist immer C-ausgeglichen, außer die Triade oder der Zyklus enthält genau eine negative Kante.

Unter der Voraussetzung der C-Ausgeglichenheit gilt das folgende Theorem.

(Gruppierbarkeits-)Theorem (Davis 1967)

Sei $G=\langle X,R\rangle$ ein S-Graph. Dann ist G C-ausgeglichen gdw es $A_1,...,A_n \subseteq X$ gibt, so daß gilt:

(1) $X = \bigcup_{i=1}^{n} A_i$

(2) $\bigcap_{i=1}^{n} A_i = \varnothing$

(3) Für alle $x,y \in X$: wenn xPy und $x \in A_i$ dann $y \in A_i$

(4) Für alle $x,y \in X$: wenn xNy und $x \in A_i$ dann $\neg\, y \in A_i$

---

[60] Man denke etwa an den Fall von drei Personen, die drei verschiedenen Gruppen angehören, und zwischen denen paarweise antagonistische Beziehungen bestehen. Diese Situation ist solange stabil, als keine Notwendigkeit zu einer Koalitionsbildung gegeben ist (Hummell/Sodeur 1984: 531).

C-ausgeglichene Strukturen (Balance im abgeschwächten Sinn) sind also so in Teilmengen zerlegbar, daß alle positiven Beziehungen sich innerhalb der Teilmenge befinden und negative Beziehungen zwischen den Teilmengen.

Das Konzept der Gruppierbarkeit bedeutet inhaltlich eine *Erweiterung von Polarisierung und Einmütigkeit*, da das Gesamtsystem in eine *Vielzahl von Teilsystemen* mit intern positiven Beziehungen zerlegbar ist. Davis konnte damit zeigen, daß die makrostrukturelle Eigenschaft der Gruppierbarkeit in einem eindeutigen Verhältnis zur Struktur der interpersonellen Beziehungen auf der Individualebene steht (Hummell/Sodeur 1984).

Ebenso wie bei der Polarisierung ist die Gruppierung aber nur eindeutig, wenn der Graph vollständig ist bezüglich positiver und negativer Kanten. In einem unvollständigen Graphen ist die Gruppierung nicht eindeutig, so daß es verschiedenene Teilmengen geben kann.[61]

Im Fall der Vollständigkeit des Graphen gilt die Äquivalenz folgender Aussagen (Davis 1967):

1. Der S-Graph ist gruppierbar (d.h. C-ausgeglichen).
2. Der S-Graph ist eindeutig gruppierbar.
3. Der S-Graph hat keinen Zyklus mit genau einer negativen Kante.
4. Der S-Graph hat keine Triade mit genau einer negativen Kante.

Bei vollständigen Graphen genügt es also, sich zur Feststellung der C-Ausgeglichenheit auf die Untersuchung von Triaden zu beschränken.

Unter Berücksichtigung der Vollständigkeit (entscheidend für die Eindeutigkeit der Gruppierung) und der beiden Balancebegriffe (entscheidend für die Zahl der Gruppen) lassen sich die Konsequenzen der Ausgeglichenheit wie folgt zusammenfassen.

Tab. 8.1: Konsequenzen von Ausgeglichenheit und Vollständigkeit (Hummell/Sodeur 1987: 144)

Vollständigkeit	*Ausgeglichenheit*	
hinsichtlich P- und N-Relation	Ausgeglichenheit im Sinn struktureller Balance (B-Balance)	Ausgeglichenheit im Sinn der Gruppierbarkeit (C-Balance)
ja	Existenz von zwei "antagonistischen" Cliquen (Polarisierung) oder Einmütigkeit	Existenz mehrerer eindeutig abgrenzbarer "antagonistischer" Cliquen
nein	Existenz von zwei Cliquen mit intern positiven oder indifferenten und extern negativen oder indifferenten Beziehungen	Existenz mehrerer Cliquen mit intern positiven oder indifferenten und extern negativen oder indifferenten Beziehungen

[61] Die metaphorische Beschreibung der Heider-Triaden bedeutet in unvollständigen Strukturen folgende Abschwächung (B-Balance) der Sätze von Kap. 6.1: (a') Der "Freund" meines "Freundes" ist nicht mein "Feind"; (b') Der "Feind" meines "Freundes" ist nicht mein "Freund"; (c') Der "Freund" meines "Feindes" ist nicht mein "Freund"; (d') Der "Feind" meines "Feindes" ist nicht mein "Feind". Im Fall der C-Ausgeglichenheit sind lediglich die Sätze (a) bis (c) (vollständige Struktur) bzw. (a') bis (c') (unvollständige Struktur) erfüllt (Hummell/Sodeur 1984: 536).

8.3.2 Gruppierbarkeit und Hierarchisierung

Der nächste Verallgemeinerungsschritt ist, daß zusätzlich zum Konzept der Gruppierbarkeit das der Hierarchisierung eingeführt wird und damit die "horizontale Dimension" um eine "vertikale Dimension" erweitert wird. Davis/Leinhardt (1972) formulieren ein Modell, das eine Situation rekonstruiert, wie sie in der Gruppensoziologie z.B. von Homans (1950) beschrieben wird: Gruppen sind häufig in der Weise strukturiert, daß sie erstens in eine Reihe hierarchischer Ebenen zerfallen und zweitens auf jeder Ebene wiederum ein Zerfallen in multiple Gruppen (Cliquen) zu beobachten ist (Hummell/Sodeur 1984: 537). Die Makro-Struktur aus teils neben- und teils über- oder untereinander liegenden Gruppen heißt "ranked clusters" und wird grafisch in Abb. 8.11 veranschaulicht.

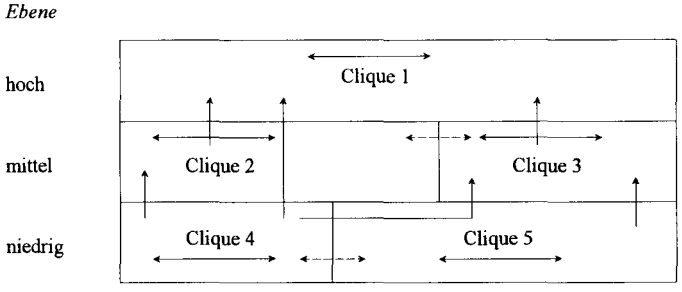


Abb. 8.11: Cliquen in hierarchischen Ebenen (nach Davis/Leinhardt 1972). Durchgezogene Doppelpfeile bedeuten beidseitige Wahlen, gestrichelte Doppelpfeile beidseitige Ablehnung und einfache Pfeile einseitige Wahlen.

Die bislang den Modellen zugrundegelegten P- und N-Relationen werden nun differenziert. Neben erwiderten positiven M-Relationen und nicht erwiderten, negativen N-Relationen gibt es nun auch einseitige A-Relationen mit positiven Verbindungen in die eine und negativen Verbindungen in die andere Richtung.

Davis/Leinhardt (1972) können zeigen, daß ein Graph als "ranked clusters" darstellbar ist, wenn die Knoten so in Teilmengen, die als Ebenen und in Teilmengen, die als Cliquen bezeichnet werden, zerlegt werden können, daß gilt:

1. Knoten gehören dann und nur dann verschiedenen Ebenen an, wenn zwischen ihnen A-Relationen bestehen;
2. Knoten befinden sich in der gleichen Clique, dann und nur dann, wenn sie durch M-Relationen verbunden sind;
3. Die Ebenen bilden eine vollständige Ordnung.

Besagt das Balanceprinzip im letzten Modell von Davis, daß über die Zeit eine Tendenz zur Aufteilung in mehrere Gruppen besteht, so wird in diesem Modell zusätzlich eine hierarchische Komponente eingeführt. Das Balanceprinzip im Modell von Davis/Leinhardt bedeutet dann, daß über die Zeit hinweg eine Tendenz zu multipler Gruppierung *und* Hierarchisierung besteht. Auf das Ranked-Clusters-Modell gehen wir nicht weiter ein, da es in der nächsten Version aufgeht.

9. Allgemeine transitive Strukturen: das T-Graph-Modell

Das Modell des transitiven Graphen - oder kurz: T-Graph-Modell - bildet einen vorläufigen Abschluß der hier verfolgten Entwicklungslinie von Balancetheorien. Ausgeglichenheit, Polarisierung, Gruppierung und Hierarchisierung wird mit einem einzigen Konzept erfaßt, dem Konzept der Transitivität. Holland/Leinhardt (1971) verwirklichen mit diesem Modell eine Zusammenfassung und Generalisierung der bisherigen Ansätze, indem sie früher behandelte Strukturaspekte als Folgen von Transitivität plus zusätzlicher Bedingungen charakterisieren. Was Heider Balance nennt, ist allgemein Transitivität.

9.1 Gruppierung und Hierarchisierung als Konsequenz von Transitivität

Wir stellen das Modell wieder in einer strukturalistischen Deutung dar. Da die Rekonstruktion völlig analog zu der Heider- und Abelson-Rosenberg-Theorie erfolgt, verzichten wir weitgehend auf Kommentare zu den strukturalistischen Prädikaten. Die Grundbegriffe und -relationen sind wieder sehr einfach: es gibt lediglich eine endliche, nicht leere Menge X und eine zweistellige, gerichtete Relation R auf X. $\langle X,R\rangle$ definiert dann äquivalent einen (unbewerteten) Digraphen. Wir interpretieren X zunächst als eine Menge von Personen und R als Liking- oder Wahl-Relation.

Definition 1
x ist ein *Holland-Leinhardt-Graph* ($x \in HLG$) genau dann, wenn es X und R gibt, so daß gilt:
(1) $x= \langle X,R\rangle$
(2) X ist eine endliche, nicht-leere Menge
(3) $R \subseteq X \times X$

Auf der Basis der Relation R lassen sich drei neue Relationen definieren, indem für alle Knotenpaare festgestellt wird, ob die Wahl erwidert wird, einseitig ist oder überhaupt keine Wahl existiert.

Definition 2
Wenn $x= \langle X,R\rangle \in HLG$, dann gilt für alle $x, y \in X$:
(1) xMy gdw xRy und yRx
(2) xAy gdw xRy und $\neg$ yRx
(3) xNy gdw $\neg$ xRy und $\neg$ yRx

M bezeichnet also die gegenseitige ("mutual") Wahl, A die einseitige ("asymmetric") Wahl und N die beidseitige Nicht-Wahl.

Mit Hilfe dieser drei Relationen versucht das Modell, die obengenannten Phänomene in Gruppen zu präzisieren (Hummell/Sodeur 1984: 538):

1. Von A-Beziehungen wird angenommen, daß sie Personen auf verschiedenen Ebenen verbinden, während M- und N-Beziehungen Personen auf der gleichen Ebene verbinden. Für Paare, die durch A verbunden sind, wird vorausgesetzt, daß der Empfänger der A-Beziehung der höheren Ebene angehört.
2. Von M-Beziehungen wird angenommen, daß sie innerhalb einer hierarchischen Ebene Personen verbinden, die der gleichen Teilgruppe (Clique) angehören. N-Beziehungen "verbinden" innerhalb einer Ebene Personen aus verschiedenen Cliquen.

Damit sich diese Hierarchisierung mit Gruppierung nach Abb. 8.11 (S.179) ergibt, muß der Graph aber transitiv sein. Der Begriff der "Transitivität" löst nun den Begriff der "Balance" ab. Die Transitivitätsforderung wird in Definition 3 festgelegt.

Definition 3
Wenn $x = \langle X,R \rangle \in$ HLG, dann gilt:
x ist ein transitiver Graph ($x \in$ T-Graph) gdw für alle $x,y,z \in X$ gilt:
(1) xRx (Reflexivität)
(2) wenn xRy und yRz dann xRz (Transitivität)

Ein transitiver Graph ist also dadurch charakterisiert, daß die dem Graphen unterliegende Relation R reflexiv und transitiv ist.[62] (2) ist die zentrale Forderung, welche - inhaltlich betrachtet - besagt, daß, wenn a b wählt und b c wählt, auch a c wählen muß.

Der Zusammenhang zwischen Transitivität und Balance wird explizit bereits von Heider (1946, 1977) erwähnt. In seinem ersten Artikel charakterisiert er beispielsweise L- und U-Relationen als psychologisch transitiv: "Logically, L is not transitive but there exists a psychological tendency to make it transitive when implications between U relations do not interfere with transitivity. The relation U, too, seems to be in this sense psychologically transitive" (Heider 1946: 109-110).

Ist Definition 3 erfüllt, so lassen sich einige z.T. interessante Eigenschaften zeigen. Das folgende Theorem faßt einige leicht beweisbare Charakteristika von M, N und A zusammen.

Theorem 1
Wenn $x = \langle X,R \rangle \in$ HLG und $x \in$ T-Graph, dann gilt für alle $x,y,z \in X$:
(1) xMx
(2) wenn xMy dann yMx
(3) wenn xMy und yMz dann xMz
(4) wenn xAy dann $\neg$ yAx
(5) wenn xAy und yAz dann xAz
(6) $\neg$ xNx
(7) wenn xNy dann yNx

Beweise stehen nicht im Originalpapier und werden hier nachgeholt.

---

[62] Die Bedingung der Reflexivität ist nur Konvention um triviale Ausnahmen zu vermeiden.

Seien die Voraussetzungen von Theorem 1 erfüllt (für die Behauptung (2), (4) und (7) von Theorem 1 muß $\langle X,R\rangle$ kein T-Graph sein).

(1) xMx folgt unmittelbar aus xRx.

(2) Angenommen xMy, dann xRy und yRx, also: yMx.

(3) Angenommen xMy und yMz, dann xRy und yRx und yRz und zRy, dann xRz und zRx (mit Transitivität), also: xMz.

(4) Angenommen (i) xAy und angenommen es gilt (ii): yAx. Dann (aus i) $\neg$yRx und (aus ii) yRx, also ist (ii) widerlegt und es gilt: $\neg$yAx.

(5) Angenommen (i) xAy und (ii) yAz. Dann (aus i) xRy und $\neg$yRx und (aus ii) yRz und $\neg$zRy. Zu zeigen ist: xAz, also (iii) xRz und (iv) $\neg$zRx. xRz folgt mit Transitivität aus xRy und yRz. Angenommen zRx, dann mit Transitivität zRy; in Widerspruch zu $\neg$zRy. Also ist die Annahme zRx widerlegt und (iv) gültig.

(6) Angenommen xNx, dann $\neg$xRx, in Widerspruch zu Reflexivität von R, also $\neg$xNx.

(7) Folgt unmittelbar aus Def. 2 (3) □

Die Relationen M, A und N haben zusammengefaßt folgende Eigenschaften:

- M ist reflexiv (1), symmetrisch (2) und transitiv (3);
- A ist asymmetrisch (4) und transitiv (5);
- N ist irreflexiv (6) und symmetrisch (7).

Da M reflexiv, symmetrisch und transitiv ist, ist M eine Äquivalenzrelation auf X. M partitioniert deshalb X in disjunkte Teilmengen mit der Eigenschaft, daß x und y in der gleichen Teilmenge sind genau dann, wenn xMy gilt. Diese Teilmengen können als Cliquen interpretiert werden. Liegen M-Beziehungen vor, dann zerfällt X also in Cliquen, die im folgenden als M-Cliquen bezeichnet werden.

Theorem 2 (Strukturtheorem, Holland/Leinhardt 1971)
Wenn $x = \langle X,R\rangle \in$ HLG und $x \in$ T-Graph, dann gibt es Teilmengen (M-Cliquen) $U_i \subseteq X$ (i=1,...,n), so daß gilt:

(1) $\bigcup_{i=1}^{n} U_i = X$

(2) $\bigcap_{i=1}^{n} U_i = \varnothing$

(3) Für alle U_i und für alle $x,y \in X$: $x,y \in U_i$ gdw xMy

(4) Für alle U_i, U_j mit $i \neq j$ und für alle $x \in U_i$, $y \in U_j$ gilt: entweder xNy oder xAy oder yAx

Theorem 2 besagt, daß in einem transitiven Graphen die Knotenmenge X in M-Cliquen aufgeteilt werden kann mit folgenden Eigenschaften:

- X ist eine Partition (1 und 2);
- innerhalb jeder M-Clique sind alle Paare von Individuen durch M-Kanten verbunden (3);
- zwischen zwei verschiedenen M-Cliquen sind alle Paare von Individuen entweder verbunden durch A-Kanten in der gleichen Richtung oder durch N-Kanten (4).

Das nächste Theorem sagt aus, wie M-Cliquen mit A und N zusammenhängen.

Theorem 3
Wenn $x = \langle X,R\rangle \in$ HLG, $x \in$ T-Graph, U eine M-Clique und $u, v \in U$, dann gilt für alle $x \in X$:
(1) uAx gdw vAx
(2) xAu gdw xAv
(3) uNx gdw vNx

Beweis von (1) unter den gegebenen Voraussetzungen:
Von links nach rechts:
Angenommen uMv und uAx, dann vRu und uRx, dann vRx (mit Transitivität).
Beweis von $\neg$xRv durch Widerspruch: Angenommen xRv, dann xRv und vRu, dann xRu (Trans), in Widerspruch zur Annahme uAx ($\neg$xRu), also $\neg$xRv.
Also vRx und $\neg$xRv, also vAx.
Von rechts nach links analog.
(2) (3) analog □

Inhaltlich besagt Theorem 3, daß in einer M-Clique alle Mitglieder strukturell äquivalent sind in dem Sinn, daß alle Personen einer M-Clique in gleicher Relation zu allen anderen in der Struktur stehen. Wählt ein Mitglied einer Clique eine Person außerhalb der Clique, gehen alle anderen Mitglieder ebenfalls zu dieser Person eine Relation ein. Bricht umgekehrt ein Cliquen-Mitglied die Relation zu einem Nicht-Cliquen-Mitglied ab, beenden alle anderen Cliquen-Mitglieder die Relation ebenfalls.
Die Hauptimplikation von Theorem 2 ist, daß auf der Grundlage der Beziehungen zwischen Personen auch Beziehungen zwischen M-Cliquen definiert werden können. Definition 4 führt eine Ordnungsrelation A* auf den M-Cliquen selbst ein.

Definition 4
Wenn $x = \langle X,R\rangle \in$ HLG, $x \in$ T-Graph, U, V zwei M-Cliquen von X sind, dann gelte U A* V gdw:
(1) uAv für alle $u \in U$ und $v \in V$
oder
(2) $U = V$

A* ist reflexiv per definitionem. Aus Theorem 1-4 und 1-5 folgt, daß A* anti-symmetrisch und transitiv ist. Reflexive, antisymmetrische und transitive Relationen sind eine partielle Ordnung, d.h. A* ist eine partielle Ordnung der Menge aller M-Cliquen von X. Man erhält damit das folgende Theorem.

Theorem 4
Wenn $x = \langle X,R\rangle \in$ HLG, $x \in$ T-Graph, so bildet A* eine partielle Ordnung auf den M-Cliquen.

Die Theoreme 2 und 4 zusammengefaßt bilden die zentralen Aussagen des Modells vom transitiven Graphen.

Theorem 2 und 4

Wenn $x = \langle X,R \rangle \in HLG$, $x \in$ T-Graph, dann kann X in M-Cliquen partitioniert werden so daß gilt:

(1) innerhalb jeder M-Clique sind alle Paare von Individuen durch M-Kanten verbunden;

(2) zwischen zwei verschiedenen M-Cliquen sind alle Paare von Individuen entweder verbunden durch A-Kanten in der gleichen Richtung oder durch N-Kanten;

(3) die M-Cliquen bilden unter A* eine partielle Ordnung.

Jeder Teil dieses Theorems hat eine einfache soziometrische Interpretation. In (1) wird die interne Struktur jeder M-Clique durch gegenseitige positive Wahlen für jedes Mitgliedspaar gekennzeichnet. (2) charakterisiert die Relationen zwischen Paaren von M-Cliquen entweder durch eine Statusordnung von einer Clique über die andere (A-Kanten) oder durch keine Statusordnung. Teil (3) schließlich charakterisiert das ganze System von M-Cliquen als konsistente Struktur im Sinn einer partiellen Ordnung.

Abb. 9.1 zeigt zwei Beispiele für solche Strukturen: Auf der linken Seite befinden sich zwei ungeordnete M-Cliquen, auf der rechten Seite eine teilweise durch A*-geordnete, hierarchische Struktur von M-Cliquen.

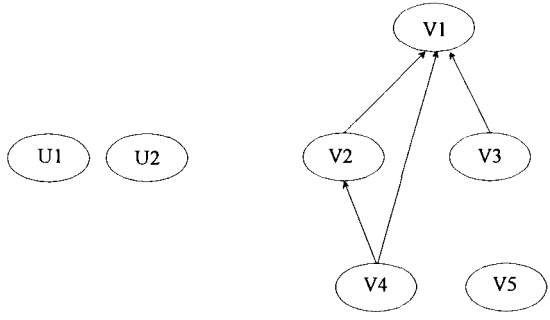


Abb. 9.1: Beispiele für geordnete und ungeordnete Cliquen. Die Ellipsen repräsentieren M-Cliquen, die Pfeile die A*-Relation

Das T-Graph-Modell zeigt somit, daß in einem Graphen ohne intransitive Wahlen *notwendigerweise* Muster von hierarchisch geordneten M-Cliquen entstehen. Zwischen psychologischen (Balance-)Zuständen von Individuen und dadurch bedingten Implikationen für die soziologische Gruppenstruktur besteht ein eindrucksvoller Mikro-Makro-Link (Hallinan/Felmlee 1975: 196).

9.2 Spezielle Fälle von T-Graphen

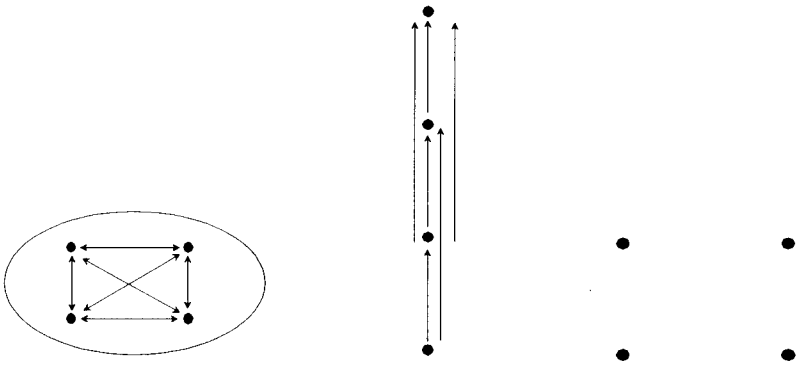
Man kann nun T-Graphen spezialisieren auf Strukturen, die weitere Bedingungen neben der Transitivität erfüllen. Diese Spezialisierungen des transitiven Graphen führen zu den historisch älteren Modellen. Auf den ersten Blick besteht zwischen älteren Modellvarianten und dem T-Graph-Modell eine begriffliche Disparität. Beispielsweise sind die Modelle von Cartwright/Harary (1956) und Davis (1967) als bewertete statt gerichtete Graphen wie hier formuliert. *Ein bewerteter Graph kann aber als Spezialfall eines gerichteten Graphen betrachtet werden, wenn positive Relationen mit M und negative mit N identifiziert werden und A leer ist* (Holland/Leinhardt 1971: 115).

Eine erste mögliche Restriktion ist, daß man die *Kantentypen* beschränkt und die sich daraus ergebenden Formationen betrachtet. Eine zweite Restriktion ergibt sich aus der Einschränkung der *Triadentypen*.

Betrachten wir zunächst Einschränkungen der Kanten. Wenn sich alle Kanten des T-Graphen aus genau *einem* der Relationentypen M, A oder N zusammensetzen, dann ist die Struktur des Graphen vollständig determiniert.

- Wenn nur M-Relationen vorliegen, dann besteht der Graph aus einer einzigen, vollständig verbundenen M-Clique. Dies ist der Sonderfall "Einmütigkeit" des Theorems von Cartwright/Harary (1956).
- Wenn nur A-Relationen vorliegen, dann besteht eine vollständige lineare Ordnung aller Individuuen.
- Wenn nur N-Relationen vorliegen, dann ist der Graph völlig unverbunden.

Abb. 9.2 veranschaulicht diese Strukturen grafisch.



Vollständige Verbundenheit: Nur M | Reihenordnung von Personen: Nur A | Völlige Unverbundenheit: Nur N

Abb. 9.2: Spezialfälle transitiver Strukturen mit genau einer der Kanten M-, A- oder N

Interessantere Möglichkeiten ergeben sich, wenn genau zwei der drei möglichen Relationen M, A oder N erlaubt sind.

- Wenn nur M- und A-Relationen vorliegen, dann bildet der T-Graph eine Quasi-Reihe.
- Wenn nur M- und N-Relationen vorliegen, dann sind alle Individuen jeder M-Clique durch M-Relationen verbunden und je zwei verschiedene M-Cliquen durch N-Relationen. Dieses Corrolar beschreibt den gruppierbaren Graph von Davis (1967).
- Wenn nur A- und N-Relationen vorliegen, gibt es eine partielle Ordnung von Personen.

Abb. 9.3 zeigt diese Strukturen grafisch.

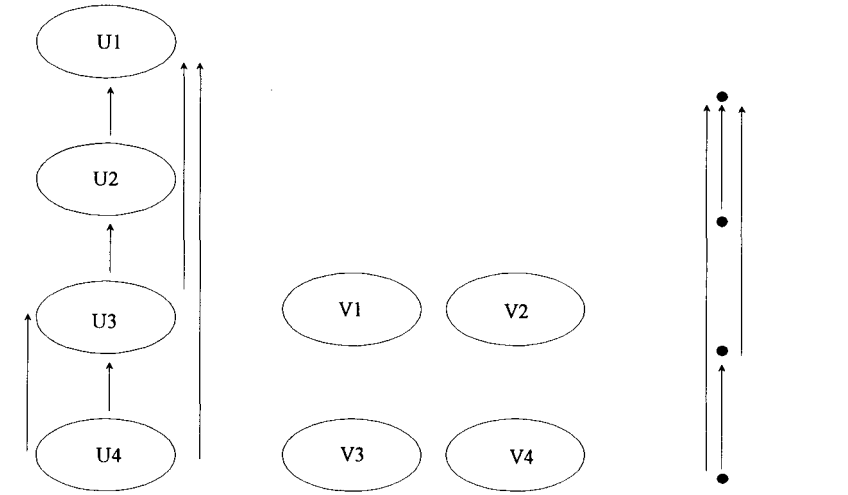


Abb. 9.3: Spezialfälle transitiver Strukturen mit genau zwei der drei Kantentypen M, A oder N

Die folgende Tabelle faßt alle Spezialfälle nochmals zusammen.

Tab. 9.1: Mögliche Graphentypen, die sich aus Restriktionen der Relationen ergeben (vgl. Holland/Leinhardt 1971: 117 und Hummell/Sodeur 1987: 147).

Zugelassene Relationen	*Resultierende Struktur*
Nur M	Vollständig verbundener Graph ("Einmütigkeit")
Nur A	Reihenordnung der Personen ("transitive tournament")
Nur N	Vollständig unverbundener Graph
Nur M, A	Quasi-Reihe (Hierarchie von Cliquen)
Nur M, N	Gruppierbarer (clusterable) Graph
Nur A, N	Partielle Ordnung (von Personen)
M, A, N	Partielle Ordnung (von M-Cliquen)

Die Strukturen, die aufgrund von Kantenrestriktionen entstehen, können alternativ auch durch das Vorkommen oder Nicht-Vorkommen entsprechender *Triadentypen* charakterisiert werden.

Eine Triade in einem Graphen mit n > 3 Knotenelementen ist ein Sub-Graph aus drei Knotenelementen, unter Eliminierung aller Relationen zu den restlichen Knoten. Insgesamt gibt es in einem Graphen mit n Knoten

$$\frac{n \cdot (n-1) \cdot (n-2)}{6}$$

Triaden. Die Triade spielt in diesem Modell insofern wieder eine Rolle, als zur Prüfung der Transitivität eines Graphen *alle möglichen Triaden* geprüft werden müssen. Eine intransitive Triade liegt vor, wenn für mindestens eines der sechs möglichen 3-Tupel ⟨a,b,c⟩ gilt: aRb, bRc und ¬aRc. Ist hingegen eine der Konfigurationen
aRb und ¬bRc, oder
¬aRb und bRc, oder
¬aRb und ¬bRc
gegeben, so ist die wenn-Komponente von Definition 3 (2) nicht erfüllt. In diesem Fall kann sowohl aRc als auch ¬aRc gelten, ohne daß die Transitivität verletzt wird. Holland/Leinhardt (1971: 117, 1970: 495) bezeichnen eine solche Triade als "leer oder neutral transitiv" ("vacously transitivity").

Insgesamt gibt es 16 verschiedene Klassen von strukturell unterscheidbaren (nicht isomorphen) transitiven, intransitiven und "leer-transitiven" Triaden, die in Abb. 9.4 aufgelistet sind. Jede Triade wird nach einem Nomenklatursystem der Form M-A-Nx mit drei Ziffern und eventuell einem Buchstaben benannt. Die drei Ziffern beziehen sich auf die Häufigkeit von M-, A- und N-Kanten (in dieser Reihenfolge), die Buchstaben D, T, U und C unterscheiden die Triaden weiter (für nähere Details vgl. Holland/Leinhardt 1970).

Ein Graph ist transitiv genau dann, wenn er keine intransitiven Triaden enthält, also keine Triaden auf der rechten Seite der Grafik in Abb. 9.4. Ein transitiver Graph, der eine partielle Ordnung von Cliquen erzeugt, enthält also ausschließlich Triadentypen der linken Seite von Abb. 9.4.

transitive und leer-transitive Triaden *intransitive Triaden*

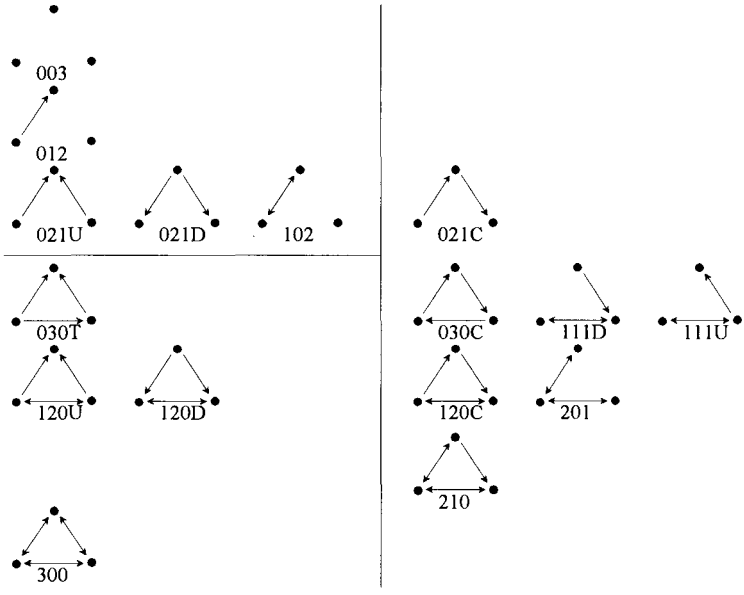


Abb. 9.4: Alle 16 strukturell unterscheidbaren Triadentypen in M (durchgezogene Doppelpfeile), A (einfache Pfeile) und N-(keine Pfeile) Notation. Die erste Zahl bezieht sich auf die Häufigkeit der M-Kanten, die zweite auf die Häufigkeit der A-Kanten und die dritte auf die Häufigkeit der N-Kanten. Die Buchstaben dienen zur weiteren Unterscheidung. Intransitive Triaden befinden sich rechts neben der vertikalen Linie, transitive Triaden unter und leer transitive Triaden über der horizontalen Linie (nach Holland/Leinhardt 1971).

Restriktionen der zugelassenen - transitiven und leer-transitiven - Triaden liefern nun als Spezialfälle die Modelle von Cartwright/Harary (1956), Davis (1967) und Davis/Leinhardt (1972).

Ein Graph ist *B-balanciert* (d.h. Einmütigkeit oder in zwei Cliquen geteilt) bei *Cartwright/Harary (1956)* genau dann, wenn es keine negativen Dreier-Zyklen gibt. Da A leer sein muß und mithin Triaden mit A-Kanten nicht zulässig sind, verbleiben als potentielle Kandidaten mit positiven und negativen Zykluswerten die Triaden 003, 102, 300 und 201. In den Triaden 003 und 201 ist der Zykluswert negativ, so daß diese auszuschließen sind. Die Menge der möglichen Triadentypen bei B-Balance ist somit auf *maximal zwei* begrenzt: *Einmütigkeit* liegt vor, wenn nur *Triaden vom Typ 300* vorkommen und *Polarisierung*, wenn es ausschließlich *Triaden vom Typ 102 und 300* gibt. Äquivalent erhält man B-Balance im Sinn von Cartwright/Harary, wenn A leer ist, R transitiv sowie eine weitere Bedingung hinzukommt, die die Zahl möglicher M-Cliquen auf höchstens zwei beschränkt (Holland/Leinhardt 1971: 119).

Man erhält einen *gruppierbaren, C-balancierten Graph* im Sinn von *Davis (1967)* wenn R transitiv und A leer ist. Bezogen auf Triadentypen bedeutet dies, daß *nur Triaden vom Typ 003, 102 und 300* vorkommen dürfen. Die Menge erlaubter Triaden

wird bei C-Balance im Vergleich zu B-Balance lediglich um die (bei Heider als unbestimmt eingestufte) Triade 003 erweitert.

Das *Ranked-Clustering-Modell (Davis/Leinhardt 1972)* schließlich, welches Hierarchisierung und Gruppierung verbindet, liegt immer dann vor, wenn R transitiv ist und die *neutrale Triade 012 nicht auftritt* (Beweis in Holland/Leinhardt 1971). Somit sind die Triaden 003, 021U, 021D, 102, 030T, 120U, 120D und 300 erlaubt. Der einzige Unterschied zum allgemeineren Modell mit Ordnung von M-Cliquen besteht also im Fehlen des Triadentyps 012 (Hummell/Sodeur 1987: 148).

Die verschiedenen Spezialisierungen des Modells sind im folgenden Schema zusammengefaßt.

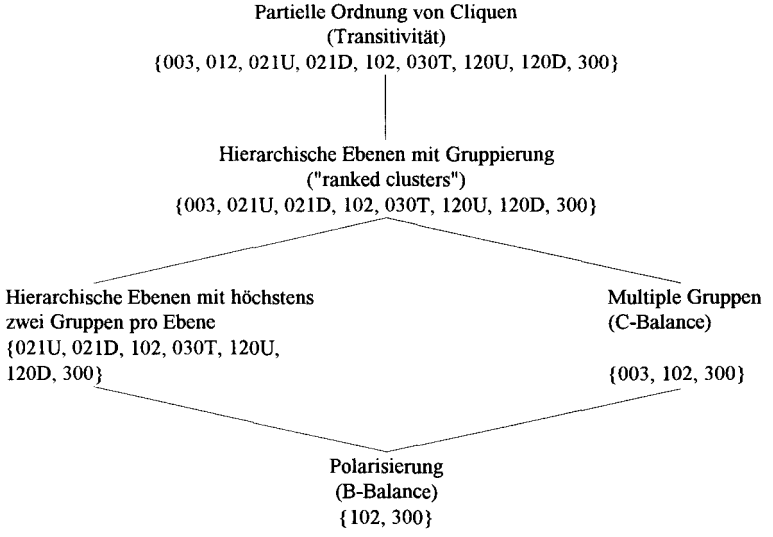


Abb. 9.5: Spezialisierungen des transitiven Graphen mit den jeweils zulässigen Triadentypen (nach Hummell/Sodeur 1987: 150)

9.3 Theoriekern und intendierte Anwendungen

Der Originalartikel von Holland/Leinhardt (1971) beschränkt sich weitgehend auf den vollständig *transitiven* Graphen mit der Analyse seiner Konsequenzen Gruppierung und Hierarchisierung, sowie auf die Diskussion der Spezialfälle von transitiven Graphen. Das Balanceprinizp und die darauf basierenden Verbindungen zu den anderen Balancetheorien werden nicht explizit angegeben.

Das Balanceprinzip der bisherigen Theorien war, daß empirische Systeme zum Gleichgewicht tendieren. "Balancetendenz" bedeutet zum einen, daß wir wie bei HT und ART wieder einen Zeitindex einführen werden, zum andern, daß analog zu Cartwright/Harary eine Unterscheidungsmöglichkeit zwischen balancierteren und weniger balancierten Strukturen gegeben sein muß. Letzteres impliziert für HL-Graphen eine Differenzierung zwischen "transitiveren" (=balancierteren) und "weniger

transitiven" (= weniger balancierten) Graphen. Das Balanceprinzip von Heider, angewendet auf diese Theorie, besagt dann, daß in Modellen der Holland-Leinhardt-Theorie eine *Tendenz zu transitiven Relationen* besteht.

Wir beziehen zunächst in der Definition des potentiellen Modells den HL-Graphen wieder auf die Zeit. Die Definitionen 5 und 6 sind völlig analog zu Heider und brauchen nicht kommentiert zu werden. Für die theoretischen Terme gelten wieder die gleichen Bemerkungen wie bei den Vorläufern: keiner der Terme ist HL-theoretisch, so daß auf die Definition des partiellen potentiellen Modells verzichtet werden kann.

Definition 5
x ist ein *potentielles Modell* der Holland-Leinhardt-Theorie ($x \in M_p(HLT)$) gdw es X, T, <, R gibt, so daß gilt:
(1) $x = \langle X, T, <, R \rangle$
(2) X ist eine endliche, nicht-leere Menge
(3) $\langle T, < \rangle$ ist eine endliche, lineare Ordnung
(4) $R: T \rightarrow Pot(X \times X)$
(5) Für alle $t \in T$: $\langle X, R(t) \rangle \in HLG$

Definition 6
$x(t) := \langle X, R(t) \rangle$

Da wir transitivere und weniger transitive Graphen unterscheiden müssen, brauchen wir einen Index, der Informationen über das Ausmaß von Transitivität in einer gegebenen Struktur gibt. Holland/Leinhardt (1970) und Davis (1970) stellen einen komplexeren Transitivitätsindex vor, der auf einem Wahrscheinlichkeitsmodell beruht und zum empirischen Modelltest verwendet wurde (vgl. unten). Für unsere Zwecke genügt ein einfacher Index, der die Abweichung einer empirischen Struktur von einem deterministischen Modell kompletter Transitivität bestimmt, wie er etwa von Hallinan/Felmlee (1975) vorgeschlagen wird.[63] Dieser Balance-Index läßt sich durch den Vergleich der Zahl intransitiver Tripel mit der Gesamtzahl aller möglichen Tripel konstruieren. Der Balance-Index soll bei uns - anders als bei Hallinan/Felmlee - maximal sein, wenn es keine intransitiven Tripel gibt, er ist minimal, wenn es ausschließlich intransitive Tripel gibt.[64] Ist x der HL-Graph, dann kann der Transitivitätsindex TRX(x) damit bestimmt werden als:
1 - Anzahl aller intransitiven Triaden / Anzahl aller möglichen Triaden.
Die Anzahl aller möglichen Triaden wird hierbei nach der in Kapitel 9.2 genannten Formel ermittelt.

---

[63] Vor- und Nachteile der verschiedenen Indizes werden diskutiert in Hallinan/Felmlee (1975).

[64] Einfacher ließe sich der Transitivitätsindex über die Zahl *transitiver* Triaden definieren. In Hinblick auf Effizienzüberlegungen für die spätere Rechnermodellierung wurde jedoch die Definition über die Anzahl *intransitiver* Triaden gewählt.

Definition 7
Wenn $x = \langle X,R\rangle \in M_p(HLT)$, dann werden n und TRX(x) wie folgt definiert:
(1) n=card(X)

$$(2)\ TRX(x) := 1 - \frac{card\{\langle x,y,z\rangle |\ x,y,z \in X \wedge xRy \wedge yRz \wedge \neg xRz\} \cdot 6}{n \cdot (n-1) \cdot (n-2)}$$

Das *Modell* der Holland-Leinhardt-Theorie läßt sich analog wie bei HT und ART definieren mit der Variante, daß das eigentliche inhaltliche Axiom (3) auf den Balance-Index Bezug nimmt.

Definition 8
x ist ein *Modell* von HLT ($x \in M(HLT)$) gdw es X, T, <, R gibt, so daß gilt:
(1) $x = \langle X,T,<, R\rangle$
(2) $x \in M_p(HLT)$
(3) Für alle t, t' ∈ T: wenn $t < t'$ dann $TRX(x_t) \leq TRX(x_{t'})$

Das Fundamentalgesetz (3) drückt die "Tendenz zu Transitivität" aus. Es besagt, daß für zwei beliebige Zeitpunkte t und t' mit t' größer t der Transitivitätsindex zu t' größer ist als der Index für den früheren Zeitpunkt t oder gleich bleibt. Mit anderen Worten: Transitivität (und damit Balance) bleibt über eine Zeitperiode hinweg entweder gleich oder nimmt zu. Tabelle 9.2 veranschaulicht die Modelldefinition mit drei fiktiven Beispielen. Die Zahlen in den Tabellenzeilen sollen die Entwicklung der Transitivitätsindizes über sechs Zeitpunkte repräsentieren. Nach der Modelldefinition sind die potentiellen Modelle S1 und S2 Modelle von HLT, S3 ist hingegen kein Modell von HLT.

Tab. 9.2: Drei Beispiele für mögliche zeitliche Entwicklungen der Transitivität. Die potentiellen Modelle S1 und S2 sind Modelle, S3 ist kein Modell von HLT

	Zeit					
M_p	t_1	t_2	t_3	t_4	t_5	t_6
S1	0,3	0,3	0,5	0,6	0,6	0,7
S2	0,6	0,6	0,6	0,6	0,6	0,6
S3	0,4	0,2	0,2	0,4	0,4	0,5

Es sei an dieser Stelle darauf hingewiesen, daß unsere Rekonstruktion des Balancegesetzes liberaler ist als die von Stephan (1990) bei der Cartwright-Harary-Theorie. Nach unserer Fassung muß der Balancegrad für ein bestimmtes Zeitintervall *nicht* zunehmen, sondern kann auch gleich bleiben (vgl. S2 von Tab. 9.2). Wir befinden uns damit in Einklang mit der Interpretation von Sukale (1971). Stephan (1990: 74) hingegen stellt die schärfere Forderung auf, daß unbalancierte Systeme immer einen Gleichgewichtszustand anstreben (was im übrigen auch mit einer Verkomplizierung des inhaltlichen Axioms verknüpft ist).

Wir kommen damit zu den intendierten Anwendungen der HL-Theorie, deren Problematik hier etwas ausführlicher dargestellt werden soll. Es werden zunächst drei

empirische Untersuchungen vorgestellt, die mit den von Holland/Leinhardt vorgeschlagenen Anwendungen durchgeführt wurden. Anschließend werden Extensionen der intendierten Systeme auf andere soziale Phänomene diskutiert.

Holland/Leinhardt (1971: 107-109) wählen Freundschaftsnetze (kleine Gruppen mit "Sentiment" (Gefühls-) Relationen) als intendierte Anwendung I_0 aus. X wird als Menge von Individuen interpretiert und R als Gefühlsrelation, welche operational in einem soziometrischen Test gemessen werden kann. xRy würde dann bedeuten, daß Person x Person y in einem soziometrischen Test wählt. Diese Menge I_0 bildet die einzige, von den Urhebern ausgezeichnete Anwendung.

Für dieses intendierte System sozialer Gruppen sind erfolgreiche Applikationen des Theoriekerns gelungen. Davis (1970) testete das T-Graph-Modell mit einer Analyse von 742 empirischen Soziomatrizen kleiner Freundschaftsgruppen. Bei diesem Test wurde für jeden Triadentyp der Prozentsatz der Soziomatrizen berechnet, bei denen die beobachtete Triadenzahl eines bestimmten Typs eine nach einem Wahrscheinlichkeitsmodell erwartete Anzahl übersteigt. Es zeigte sich, daß die kritischen intransitiven Triaden signifikant seltener und die Triaden mit transitiver Struktur häufiger auftraten als nach dem Zufallsmodell erwartet wurde. Eine weitere empirische Stützung ergab sich, als die Triaden zusätzlich differenziert wurden in mehr und weniger transitive bzw. intransitive Triaden. Beispielsweise enthält die Triade 030C drei, die Triade 201C zwei, die Triade 003 keine intransitive Tripel und die Triade 003T ein transitives und 300 sechs transitive Tupel. Man kann so eine Rangfolge von extrem intransitiven bis sehr transitiven Triaden aufstellen. Davis (1970) konnte nun nachweisen, daß der Unterschied zwischen der Anzahl der Transitivitäten und der Anzahl der Intransitivitäten ziemlich genau die beobachtete Reihenfolge voraussagt: jene Triaden mit mehr Transitivitäten wurden häufiger beobachtet als nach dem Zufallsmodell erwartet werden müßte und jene mit mehr Intransitivitäten seltener. Ein ähnlicher Test mit 51 anderen Soziomatrizen wurde mit dem gleichen Ergebnis von Hallinan (1974) durchgeführt: "The null hypothesis that friendship choices are distributed randomly ist rejected for every group, and the hypothesis of transitivity is supported. It is apparent that the structure of sentiment relations in these groups approximates the structure of the transitivity model, namely, hierarchies of ranked clusters of cliques" (Hallinan 1974: 366-368). Ein identisches Ergebnis zeigte sich bei Holland/Leinhardt (1975). Die HL-Theorie sagt strukturelle Trends für Freundschaftsnetze somit signifikant besser voraus als das Zufallsmodell.

Soziale Gruppen mit Gefühlsrelationen bilden also einen gut gesicherten Anwendungsbereich der HL-Theorie. Die Autoren geben keinen Hinweis, daß das Modell auf weitere empirische Phänomene bezogen werden könnte. Man kann sich nun - im Sinn der "paradigmatischen Methode" - Anwendungsfälle überlegen, die der von den Begründern der Theorie vorgegebenen Ausgangsmenge hinreichend ähnlich sind. Es liegt nahe, sich zunächst auf die ältere Fassung der Balancetheorie von Cartwright und Harary zu beziehen, die mit ihrer Theorie ja bereits eine Liste möglicher Erweiterungen vorgelegt haben (vgl. Kap. 8.2.2, S.176). Eine konkrete Extension wäre also z.B., X als Menge von Personen und R als Relation "... gibt Information an ..." zu interpretieren. Cliquen wären dann alle Personen, die untereinander Informationen aus-

tauschen, während bei hierarchischen Cliquen die Kommunikation nur einseitig verläuft und isolierte Cliquen untereinander keine Kommunikation pflegen. Eine zweite Anwendungserweiterung im Sinn von Cartwright/Harray (1956) wären Machtsysteme. Hierarchisch höhere Cliquen haben einen höheren Status und damit Macht über andere Cliquen. Die Konverse der Relation xRy kann deshalb als Machtrelation interpretiert werden: y hat Macht über x. Innerhalb von Cliquen wären dann die Machtverhältnisse "ausgewogen" durch gegenseitige M-Wahlen, zwischen den Cliquen bestünden Machtbeziehungen (Konverse der A*-Relation) oder keine Machtbeziehungen (N-Relation). Eine dritte, von Cartwright und Harary ursprünglich nicht genante Extension der Menge I_0 wäre, X als Menge von Organisationen (z.B. Firmen) zu interpretieren und R als irgendeine Beziehung zwischen einem Organisationenpaar, z.B. "... hat Handelsbeziehungen mit ...". Eine "Clique von Organisationen" wären danach Firmen, die alle untereinander Handelsbeziehungen unterhalten. Cliquen unterschiedlicher Ebenen entsprächen analog Organisationsclustern, welche einseitig Handelsbeziehungen mit Organisationen höherer Ebene unterhalten.

Die folgende Tabelle gibt einen Überblick über mögliche intendierte Anwendungen für alle hier behandelten Balancetheorien, wie sie in der Literatur diskutiert werden. (C) und (D) gehen dabei über die ursprünglichen Vorschläge von Cartwright/Harary (1956) hinaus. HT und ART beschränken sich auf (A), während alle anderen Theorien und insbesondere HLT im Prinzip auf *alle* Beispiele (A)-(D) angesetzt werden können.

Tab. 9.3: Intendierte Anwendungen für Balancetheorien (nach Cartwright/Harary 1979: 27)

Knoten	*Kanten*
(A) Kognitive Elemente	Kognitive Relationen
Personen	Gefühls- und Einheitsrelationen
Einstellungsobjekte	
Propositionen	
Eigenschaften	
(B) Personen	Interpersonelle Relationen
	Soziometrische Wahlen
	Kommunikation
	Interaktion: Helfen, schenken etc.
	Macht und Einfluß
(C) Gruppen, Organisationen	Intergruppen Relationen
	Bündnis/Kriege
(D) Nationen	Internationale Relationen
	Bündnis/Kriege
	Handel

Verschiedene Autoren bestreiten die Anwendbarkeit der HL- bzw. Balancetheorien auf die meisten der in Tab. 9.3 aufgelisteten Phänomene. Wir möchten die Diskussion in einigen zentralen Punkten wiedergeben.

Anderson (1979) diskutiert die Relevanz der Balancetheorien für die allgemeine Analyse sozialer Netzwerke. Anderson behauptet, daß Transitivitätstendenzen zwar für kleine Freundschaftsgruppen zutreffen mögen, in institutionellen Kontexten mit

instrumentellen Transaktionen sei dieses Prinzip aber nicht brauchbar. Unbalancierte Triaden sind beispielsweise in der Politik sehr gebräuchlich und stabil und Balance erscheint dort weitaus weniger zwingend als in Freundschaftsnetzen. Ist beispielsweise (eine Nation) a mit b politisch verbunden und b mit c, dann kann a mit c politisch langfristig alliiert sein *oder auch nicht*. In einem anderen Kontext kann a ebenso ausgezeichnete und stabile Relationen sowohl zu b und c unterhalten, obwohl b und c politische Feinde sind.

Ein Politiker, dessen Macht und Einfluß auf einem Netz von Verbündeten basiert, kann sich - so Anderson - nicht oft Balancepraktiken gönnen. Anderson schlägt vor, Balance einschließlich des Transitivitätsprinzips nicht als Verhaltensgesetze, sondern vielmehr als Daumenregeln aufzufassen, die Personen in bestimmten Kontexten anwenden, wenn sie ihre Lebenswelt konstruieren. Diese Regeln werden von bestimmten Kontextmarken ausgelöst und Aufgabe der Balancetheoretiker wäre es, diese Kontextmarken genau anzugeben (Anderson 1979: 461). Davis/Holland/Leinhardt haben eine Menge solcher Gruppensituationen gefunden, in denen die Teilnehmer anscheinend die Transitivitätsregel benutzen. Was Anderson vollständig vermißt, ist die Definition des Bereichs dieser Menge. Solange wir nicht genau wissen, welche Eigenschaften von Situationen Personen veranlassen, die Transitivitätsregel anzuwenden, haben wir nicht viel gelernt von der Transitivität. Die kritische und unbeantwortete Frage ist: "Precisly what kind(s) of groups were they studying" (Anderson 1979: 459).

Hallinan/Felmlee (1975) und Granovetter (1973, 1979) bezweifeln ebenso wie Anderson, daß Transitivität eine generelle strukturelle Eigenschaft sozialer Formationen ist. Für Granovetter ist Transitivität vielmehr eine Funktion der *Intensität der Relationen,* so daß die Anwendbarkeit der Balancetheorien auf institutionelle soziale Einheiten kaum gegeben ist. In der Politik gibt es z.B. zu viele Interessenskonflikte zwischen gegensätzlichen Gruppen und Akteuren, als daß die Tatsache eines gemeinsamen Feindes stets in gemeinsame Allianzen und Kooperationen münden würde. Während es in kleinen Gruppen relativ leicht ist, zur Balanceherstellung Einstellungen zu Personen zu verändern, überwiegt in größeren Kontexten indirekter Austausch gegenüber direktem Austausch und die negativen Relationen sind so fest in institutionelle Substrukturen eingebettet, daß sie nicht so leicht verändert werden können.

Die Untersuchungen von Hallinan/Felmlee (1975) weisen in der Tat darauf hin, daß Transitivität in erster Linie ein Organisationsprinzip von Freundschaftsnetzen ist, in denen Gefühlsrelationen bestehen. Anhand der obengenannten soziometrischen Daten wird gezeigt, daß Gefühlsrelationen (erhoben mit: "who are your best friends") weniger intransitiv sind als andere Relationen, die eher auf Kompetenz zielen (erhoben mit: "who would you like to work with on a given project?"). Weiter zeigte sich, daß die Intensität der Gefühlsrelation Auswirkungen auf Transitivität hat: Je intensiver Freundschaftsrelationen sind, um so weniger intransitiv sind sie und je loser Freundschaften sind, um so mehr Intransitivitäten bestehen. Die Autoren erklären dies damit, daß in intensiven Beziehungen Intransitivität psychologisch schmerzlicher ist als in losen Beziehungen.

Aus strukturalistischer Perspektive stellen sich diese Einwände wie folgt dar. Die Behauptung ist, daß sich eine relativ umfassende Menge $I^* \subseteq I$, die als intendierte Anwendung der HL-Theorie zur Diskussion steht, sich doch nicht als solche eignet. Diese aus I zu entfernende bzw. nicht aufzunehmende Menge I^* umfaßt in der Interpretation von Anderson, Granovetter und Hallinan/Felmlee zumindest die Mengen (C), (D) und Elemente von (B) in Tab. 9.3. Die Untersuchungen von Hallinan/Felmlee lassen vermuten, daß diese Anwendungen nicht zu I gehören. Betrachtet man die Menge I^* als bereits in I enthalten, würde dies einen empirischen Rückschritt darstellen: die intendierten Anwendungen der HL-Theorie müßten wieder eingeschränkt werden auf eine Teilmenge von (B) oder noch weiter auf die von den Urhebern der Theorie angegebene Menge I_0 von Kleingruppen mit Gefühlsrelationen. I_0 dürfte der "Kern" der Anwendungen sein, auf den die empirische Behauptung sicher zutrifft. Die empirischen Ergebnisse weisen außerdem darauf hin, daß es innerhalb der Menge I_0 weiter abgestufte Teilmengen gibt, zu denen die Theorie sehr gut paßt (intensive Verbindungen) und weniger gut paßt (lose Verbindungen). Dies bedeutet, daß die Balancetheorien um so "besser anwendbar" sind, je intensiver die Relationen sind.[65] Weitere empirische Forschung wäre jedoch nötig, um den Anwendungsbereich genauer anzugeben und die Ergebnisse von Hallinan/Felmlee zu bestätigen oder zu widerlegen. Wie beim Ausgangsmodell von Heider empfiehlt sich auch hier der vorsichtige Gebrauch der "Regel der Autodetermination". Die Regel bestimmt dann die Erweiterung der intendierten Systeme von HLT selbst, auf deren Basis eine genauere Charakterisierung der Anwendungsmenge der Holland-Leinhardt-Theorie erfolgen kann. Es liegt nahe, zunächst mit Fällen aus der Menge (B) zu beginnen, die der Menge I_0 "sehr ähnlich" sind, und dann den Theoriekern sukzessive auf immer "weniger ähnliche" Fälle anzuwenden.

9.4 Wissensbasierte Modellierung

In der Holland-Leinhardt-Theorie ist von Graphen die Rede, die in PROLOG unterschiedlich repräsentiert werden können (Bratko 1987: 242-243). Da Graphen äquivalent auch in relationaler Schreibweise darstellbar sind, besteht eine Möglichkeit darin, wie im Heider-Modell jede Kante als eine getrennte Clause zu codieren. Angewandt auf unser Beispiel mit den R-Kanten, könnte die Datenbasis also z.B. so aussehen:

```
r(a,b).
r(b,c).
r(a,d).
```

Ein zweites Verfahren besteht darin, den *ganzen* Graphen in *einer* Datenstruktur zu repräsentieren. Der Graph könnte als Paar von zwei Mengen dargestellt werden: Knoten und Kanten. Jede Kante wäre dabei ein Paar von Knoten. Bei einem gerichteten Graphen wären die Knotenpaare als *geordnete* Paare zu interpretieren, bei

[65] Unter unserer Rekonstruktion des Fundamentalgesetzes ist diese Differenzierung aber nicht möglich, da eine Anwendung entweder das Fundamentalgesetz erfüllt oder nicht.

einem ungerichteten wäre die Ordnung irrelevant. Ein gerichteter oder ungerichteter Graph könnte dann z.B. so codiert werden:

```
graph([a,b,c,d],[(a,b),(b,c),(a,d)]).
```

Ein drittes Repräsentationsmittel schließlich bestünde darin, jeden Knoten mit einer Liste seiner Nachbarknoten zu assoziieren:

```
graph([a,[b,d],b,[c],c,[],d,[]]).
```

Dies bedeutet für einen gerichteten Graphen, daß a mit b und d (gerichtet) verbunden ist, b mit c und c sowie d mit keinem Knoten.

Die zweite und dritte Repräsentationsart scheint in unserem Fall wenig angemessen. Bei einer größeren Datenmenge werden die Listen sehr lange und damit unhandlich. Wie in Kap. 4.2.2.3 erwähnt, haben PROLOG-Listen die Eigenschaft, daß sie nur vom Kopf her abgearbeitet werden können, so daß bei längeren Listen der Zugriff auf hintere Listenelemente ineffizient werden kann. Zudem ist die Länge der Listen abhängig von Interpreter und Maschinentyp limitiert. Die Darstellung jeder Relation als PROLOG-Fakt wird hingegen lediglich beschränkt durch die verfügbare Speichergröße für die Faktenbasis und der Zugriff auf die einzelnen Fakteneinträge ist über Matching und Backtracking keinen Limitierungen unterworfen.

Ein einfacher Eintrag mit `r(x,y)` in die Datenbasis erscheint aus den Gründen, die schon bei der Abelson-Rosenberg-Theorie diskutiert wurden, als nicht zweckmäßig. Wir deduzieren nämlich aus bekannten Fakten neue Fakten und möchten auch hier für jeden Fakt die Ursache seiner Gültigkeit festhalten, dabei aber gleichzeitig nicht die Stellenzahl der Prädikate künstlich erweitern.

Wir führen deshalb wie bei ART ein globales Prädikat `faktum/2` ein, in dem die Grundrelationen sowie die Ableitungen gespeichert werden.[66] Das erste Argument bildet der Fakt, das zweite Argument der Grund, warum der Fakt gilt:

```
faktum(fakt, ursache).
```

Für die Grundrelation R wählen wir als zweites Argument das Atom `empirisch`, für abgeleitete Relationen eine Liste mit der Nummer der benutzten Regel und den Ableitungsursachen als Einträgen.

```
faktum(r(a,b),empirisch).
faktum(m(a,b),[2,r(a,b),r(b,a)]).
```

Das zweite `faktum`-Prädikat ist dabei so zu verstehen, daß `m(a,b)` abgeleitet wurde mit Regel 2 und unter Verwendung von `r(a,b)` und `r(b,a)`.

Anders als bei ART und HT haben wir nun eine größere Vielfalt von deduzierbaren Fakten, wie z.B. "x ist eine M-Clique", "x ist ein T-Graph", "zwischen x und y besteht

---

[66] Die Speicherung der Fakten in einem einzigen Prädikat vereinfacht das Programm. Der Nachteil ist, daß das Programm "ineffizienter" wird, da bei jeder Suche nach einem Fakt `f` nicht das entsprechende Prädikat `f` gematcht wird, sondern das allgemeine - aus eventuell sehr vielen Einträgen bestehende - Prädikat `faktum`.

die A*Relation" etc. Die herzuleitenden Fakten werden als prädikatenlogische Atomformeln dargestellt. Beispielsweise entspricht der Fakt "x ist eine M-Clique" der Formel `m_clique(x)` mit der Menge (Liste) x als Argument. Analog entspricht "A*(x,y)" ("zwischen der Clique x und y besteht die A*-Relation") der PROLOG-Clause `a_stern(x,y)`. All diese Formeln erscheinen an erster Argumentstelle des Prädikats `faktum/2`.

Zur Ableitung der Fakten werden Regeln eingeführt, welche die Definitionen des Modells möglichst einheitlich repräsentieren sollen. Jede Definition soll in eine oder mehrere Regeln abgebildet werden. Die Regeln werden mit dem Prädikat `regel/4` codiert, das vier Argumente mit folgenden Informationen besitzt:

```
regel(regelnr,regeltyp,konklusion,ableitungsursache).
```

Im Vergleich zu den ART-Regeln benutzen wir also jetzt eine zusätzliche Argumentstelle, an welcher der Typ der Regel festgehalten wird.

Der Regelkörper besteht aus den zur Ableitung von `konklusion` nötigen Fakten. Will man z.B. Regeln einführen, welche es erlauben, die verschiedenen Triadentypen abzuleiten (Abb. 9.4, S.188), so könnte das Regelprädikat zur Ableitung des Triadentyps 120d wie folgt codiert werden.

```
regel(15,triadentyp,triadentyp([X,Y,Z],'120d'),
   [m(X,Z),a(Y,X),a(Y,Z)])
   :-
   faktum(m(X,Z),_),
   faktum(a(Y,X),_),
   faktum(a(Y,Z),_).
```

Die "Triadentyp-Regel" mit der Nr. 15 besagt, daß die Triade [X,Y,Z] vom Typ 120d ist, wenn zwischen X und Z eine M-Relation, und zwischen Y, X sowie Y, Z je eine A-Relation besteht. Die Liste des letzten Regelarguments hält mit den entsprechenden Instantiierungen immer die Ableitungsursache fest. Feuert die Regel, so wird das dritte Regelargument `triadentyp([X,Y,Z],'120d')` mit den entsprechenden Instantiierungen und der Ursachenliste der Faktenbasis hinzugefügt. Mit X=c, Y=d, Z=e wird die Faktenbasis also um folgenden Eintrag erweitert:

```
faktum(triadentyp([c,d,e],120d),[m(c,e),a(d,c),a(d,e)]).
```

Ein Nachteil der obigen Regeldarstellung ist, daß das Regelprädikat scheitert, wenn die M- und A-Relationen noch nicht in der Faktenbasis stehen, also noch nicht abgeleitet wurden, obwohl die Regel aufgrund der vorliegenden R-Relationen vielleicht feuern würde.

Wir führen deshalb wie bei ART einen einfachen Regelinterpreter ein, der dieses Manko aufheben und die Abarbeitung der Regeln übernehmen soll. Danach soll der Regelinterpreter bei Anfrage nach der Gültigkeit von Faktum x
(1) prüfen, ob x bereits in der Faktenbasis steht und, falls dies nicht der Fall ist,
(2) prüfen, ob x mit einer Regel abgeleitet werden kann.
Das Prädikat `leite_ab/1` führt diese Abarbeitung durch.

```
leite_ab(X) :-                          % X ist gueltig wenn
   faktum(X,_).                         % X ein Faktum ist
                                        % ODER
leite_ab(X) :-                          % X ist gueltig wenn
   regel(Nr,_,X,Ursach),                % X mit einer Regel abgeleitet werden kann
   schreibe_liste([
   'Mit Regel ',Nr,' abgeleitet: ',
   X,' -> Faktenbasis'
                  ]),
   asserta(faktum(X,[Nr|Ursach])). % Fuege X zur Faktenbasis hinzu
```

Schlägt das erste `leite_ab`-Prädikat fehl, ist `X` also noch nicht in der Faktenbasis, wird mit dem zweiten `leite_ab`-Prädikat versucht, eine Regel anzuwenden. Im Erfolgsfall wird die Konklusion `X` zusammen mit der Ursachenliste und der Regelnummer der Faktenbasis am Anfang (mit `asserta/1`) hinzugefügt. Dieser einfache Inferenzmechanismus wird im Verlauf der Implementierung erweitert.
Die `faktum`-Prädikate im Regelkörper können jetzt also durch `leite_ab/1` ersetzt werden, so daß Regel 15 nun folgendes Format hat:

```
regel(15,triadentyp,triadentyp([X,Y,Z],'120d'),
   [m(X,Z),a(Y,X),a(Y,Z)])
   :-
   leite_ab(m(X,Z)),
   leite_ab(a(Y,X)),
   leite_ab(a(Y,Z)).
```

Regel 15 scheitert jetzt nicht mehr, wenn M- und A-Relationen noch nicht deduziert wurden. Vielmehr werden in diesem Fall (rekursiv) die - noch zu definierenden - Regeln für M- und A-Relationen aufgerufen.

`leite_ab/1` ist damit das Grund-Prädikat zur Anfrage an die Wissensbasis und zur Herleitung neuer Fakten. In bestimmten Fällen sollen aber keine Ableitungen mit Einträgen in die Faktenbasis erfolgen, sondern es soll nur die Gütigkeit eines Fakts bewiesen werden. Hierzu soll das Prädikat `zeige/1` dienen, das analog aufgebaut ist wie `leite_ab/1` mit der einzigen Variante, daß das `asserta`-Prädikat in der zweiten Clause fehlt.

9.4.1 Grund-Prädikate zum allgemeinen T-Graphen

Wir implementieren nun Schritt für Schritt die in Kap. 9.1 bis 9.3 eingeführten Definitionen und Theoreme. Definition 1 führt die Knotenmenge X und die Relationen R ein. Die eleganteste Repräsentation von Definition 1 könnte in einem Prädikat

`graph(X,R)`

oder

`holland_leinhardt_graph(X,R)`

bestehen, wobei X die Knotenmenge und $R \subseteq X \times X$ die Relationenliste wäre. Der Aussage: "Der Graph <X,R> ist ein T-Graph" entspräche dann auf PROLOG-Ebene z.B.:

```
t_graph(graph(X,R)).
```

Wie oben angesprochen, hat die Datenrepräsentation in einer einzigen Struktur erhebliche Nachteile, so daß Knotenliste und Relationen getrennt gespeichert werden und jede Relation einem Fakt entspricht. Aussagen über den Graphen als Ganzes beziehen sich immer auf die aktuelle, fallspezifisch gegebene Knoten- und Relationenmenge.

Praktisch geschieht das Einlesen von Knoten und Relationen entweder über eine Datei oder über die Tastatur. Im Fall des Einlesens von einer Datei wird erwartet, daß zuerst die Knoten aufgelistet sind und dann alle Kanten:

```
a.
b.
c.
r(b,c).
r(b,a).
r(a,b).
```

Die Relationen werden vom Programm intern in das `faktum`-Prädikat eingetragen, die Knoten in eine Knotenliste `knoten/1` (für Details vgl. Anhang).

Im Fall des Einlesens über die Tastatur wird zuerst die Knotenmenge eingelesen, dann das Cartesische Produkt gebildet und für jedes Paar `[X,Y]` mit $X \neq Y$ das Bestehen der R-Relation abgefragt. Sind die Einleseprozeduren regulär abgearbeitet, dann gibt es in der Datenbasis Knoten und Kanten gemäß Definition 1 von HLT, so daß Definition 1 erfüllt ist.

In Definition 2 werden die von R abgeleiteten Relationen eingeführt. Definition 2 entsprechen die folgenden drei Regeln, die völlig analog übernommen werden können.

```
regel(1,man, m(X,Y),[r(X,Y),r(Y,X)]) :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,X)).

regel(2,man, a(X,Y),[r(X,Y), not, r(Y,X)]) :-
   leite_ab(r(X,Y)),
   not leite_ab(r(Y,X)).

regel(3,man, n(X,Y),[not, r(X,Y), not,r(Y,X)]) :-
   not leite_ab(r(X,Y)),
   not leite_ab(r(Y,X)).
```

Mit dem Beispielgraphen aus Abb. 9.9 (S.232) würden folgende Ableitungsergebnisse erzeugt:

`?- leite_ab(r(c,k))` liefert `true`, da `r(c,k)` bereits in der Faktenbasis steht;

`?- leite_ab(a(c,k))` liefert `true`, und `a(c,k)` wird zur Faktenbasis hinzugefügt;

`?- leite_ab(a(g,d))` liefert `fail`, und `a(g,d)` wird nicht zur Faktenbasis hinzu gefügt;

`?- leite_ab(m(X,Y))` liefert (je nach Reihenfolge der R-Relationen in der Faktenbasis) z.B. `X=a, Y=b`, und `m(a,b)` wird zur Faktenbasis hinzugefügt.

Definition 2 ist eine Allaussage, eine PROLOG-Anfrage liefert aber immer nur *ein* Ergebnis. Sollen *alle* in der Faktenbasis gültigen M-, A- oder N-Relationen abgeleitet werden, so muß das `leite_ab`-Prädikat mit einem `fail` zum Backtracking ange-

stoßen werden. Der PROLOG-Backtracking-Mechanismus übernimmt die Herleitung aller gültigen M-, A- und N-Relationen, so daß man sich als Programmierer nicht weiter darum kümmern muß.
Mit

```
?- leite_ab(m(X,Y)),fail.
```

würden somit *alle gültigen* M-Relationen hergeleitet und in die Faktenbasis eingetragen werden.

Zur Vereinfachung definieren wir für die vorliegenden und zukünftigen Regeln Prädikate, welche die Ableitung der wichtigsten Konklusionen übernehmen.[67]

```
m :-
   leite_ab(m(X,Y)),
   fail.
m.
a :-
   leite_ab(a(X,Y)),
   fail.
a.
n :-
   leite_ab(n(X,Y)),
   fail.
n.
```

Zwei Besonderheiten sind noch zu erwähnen, auf die erste haben wir oben bereits hingewiesen.

- Im Unterschied zu den Definitionen kann mit den Regeln im Programm nur von rechts nach links abgeleitet werden, nicht von links nach rechts.
- Eine Schwierigkeit ergibt sich bei Regel 3 als Folge des PROLOG-not. Falls `leite_ab(n(X,Y))` - und damit `not leite_ab(r(X,Y))` - mit Variablen aufgerufen wird (was normalerweise der Fall ist), wird `leite_ab(r(X,Y))` für eine bestimmte Bindung wahr. Damit wird `not leite_ab(r(X,Y))` falsch und die Bindung an `X` und `Y` wird gelöst. `X` und `Y` werden wieder zu freien Variablen und Backtracking setzt ein.[68] Dies wird fortgesetzt bis zum Ende der Faktenbasis. Es wird also kein Wert für `X` und `Y` gesucht, für den `leite_ab(r(X,Y))` *nicht* aus dem Programm folgt, d.h. es wird keine N-Relation hergeleitet bei Aufruf mit nichtinstantiierten Argumenten. Hierzu bieten sich zwei Lösungsalternativen an, wobei die zweite realisiert ist:
  1. Man könnte zu R eine weitere Grundrelation `NOT_R(X,Y)` einführen, die dann zutrifft, wenn zwischen X,Y keine R-Relation besteht. Ein Nachteil ist, daß die Menge der Grunddaten mit zunehmender Knotenmenge und abnehmender Dichte des Netzes größer wird.
  2. Man prüft vor dem Aufruf der Regel, ob die Argumente `X`, `Y` in `n(X,Y)` instantiiert sind. Sind die Argumente nicht instantiiert, werden sie über das Cartesische Produkt aus der Knotenmenge instantiiert. Die Instantiierungs-Prüfung

---

[67] Die zweite Clause `m/0` ohne Regelkörper (bzw. `a/0`, `n/0`) dient nur dazu, um nach vollständiger Abarbeitung der ersten Clause ein Scheitern zu verhindern.

[68] Vgl. Kleine-Büning/Schmitgen (1986: 96-97) und das dort diskutierte Beispielprogramm.

erfolgt durch das Einfügen des Prädikats `check/1` in der ersten `leite_ab`- bzw. `zeige`-Clause. Sind die Argumente nicht instantiiert, wird das Cartesische Produkt $K = X \times X$ gebildet und mit jedem Instanzen-Paar $[x,y] \in K$ `leite_ab(n(x,y))` versucht. Bei Regel 2 ergab sich das Problem nicht, da die `not`-Clause das zweite Teilziel der Regel ist und die Variablen über die erste Clause auf alle Fälle instantiiert werden.

Der transitive Graph wird in Definition 3 festgelegt. Die Bedingung der Reflexivität lassen wir hier und künftig außer acht, da sie nur für formale beweistechnische Zwecke benötigt wird. Zudem würde die Einführung der Reflexivität die Faktenbasis unnötig erweitern (und unnötigen programmiertechnischen Folgeaufwand bedeuten).

Das zentrale Prädikat des Modells ist "... ist ein T-Graph". Für die Definition des T-Graphen in PROLOG würde ein 0-stelliges Prädikat `t_graph` genügen. `t_graph` wäre wahr, wenn die rechte Seite der Äquivalenz von Definition 3 erfüllt ist, ansonsten falsch. Da später aber noch andere Eigenschaften festgestellt werden sollen, die den Graphen als Ganzes charakterisieren, führen wir ein 2-stelliges Prädikat `graph/2` ein. Das erste Argument ist der Graphentyp, das zweite Argument ein Wahrheitswert. `graph(t_graph,true)` besagt damit, daß der (Holland-Leinhardt-)Graph ein T-Graph ist, `graph(t_graph,false)`, daß es sich um keinen T-Graphen handelt. Zur besseren Behandlung als PROLOG-Regel wird Definition 3-2 logisch wie folgt umgeformt:

(D3-2') Ein Graph $\langle X,R \rangle$ ist *kein* T-Graph gdw es ein $x,y,z \in X$ gibt so daß gilt: xRy und yRz und $\neg$ xRz.

Dieser Definition entspricht in PROLOG Regel 4.

```
regel(4,graph,graph(t_graph,false),[r(X,Y), r(Y,Z), not, r(X,Z)])
   :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,Z)),
   ungleich(X,Z),
   not leite_ab(r(X,Z)).
```

Der Backtracking-Mechanismus von PROLOG sorgt dafür,

- daß die Faktenbasis entweder so lange durchsucht wird, bis das Regelprädikat erfolgreich ist, also ein intransitives Tripel gefunden ist;
- daß die Regel scheitert, falls kein intransitives Tripel gefunden wird.

Zu beachten ist bei Regel 4 die dritte Clause im Regelkörper, welche die Ungleichheit von X und Z fordert. Bei `r(a,b), r(b,a)` würde ansonsten das `leite_ab`-Prädikat in der letzten Clause mit `r(a,a)` instantiiert. Da R in der Datenbasis nicht reflexiv ist, ist `r(X,X)` immer falsch, so daß `not leite_ab(a,a))` wahr wäre und die Regel einen falschen Schluß produzieren würde. Das Beispiel zeigt, daß die Definitionen einerseits zwar ziemlich natürlich in die Programmiersprache übernommen werden können, andererseits aber auch nicht "blind-mechanisch" übertragen werden können.

Im Fall des Scheiterns von Regel 4 *muß* ein T-Graph vorliegen, was mit Regel 5 ausgedrückt wird.

```
regel(5,graph,graph(t_graph,true),
   [r(x,y),r(y,z),r(x,z), 'fuer alle Knoten x, y, z'])
   :-
   not leite_ab(graph(t_graph,false)).
```

Anders als bei den ersten drei Regeln ist in Regel 4 erstmals die Ursache für die Gültigkeit der Konklusion (`graph(t_graph,false)`) nicht trivial, da die Regel ein intransitives Tripel *entdeckt*. Wünschenswert ist natürlich, dieses bzw. alle intransitiven Tripel in der Faktenbasis festzuhalten. Hier böte sich das Prädikat `graph/2` von Regel 4 an, was allerdings zwei Nachteile hat. Erstens müßte eine weitere Argumentstelle hinzugenommen werden bzw. die Argumentstruktur anders aufgebaut werden. Will man zweitens *alle* intransitiven Tripel, so müßte `graph(t_graph,false)` mehrmals erfüllbar sein und würde damit mehrmals in der Faktenbasis stehen. Wir definieren deshalb eine eigene Regel, welche die Herleitung aller intransitiven Tripel übernimmt.

```
regel(6,transtest,triade([X,Y,Z],intransitiv),
   [r(X,Y),r(Y,Z),not,r(X,Z)])
   :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,Z)),
   ungleich(X,Z),
   not leite_ab(r(X,Z)).
```

Regel 7 prüft analog auf echt transitive Triaden bzw. liefert bei vollständigem Backtracking alle echt transitiven Triaden. Da reflexive Relationen in der Datenbasis nicht zugelassen sind, entfällt hier die Notwendigkeit der Prüfung auf Ungleichheit.

```
regel(7,transtest,triade([X,Y,Z],transitiv),[r(X,Y),r(Y,Z),r(X,Z)])
   :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,Z)),
   leite_ab(r(X,Z)).
```

Bislang wurden nur Definitionen behandelt. Zweck von Definitionen ist es, mit "grundlegenden" Prädikaten (dem Definiens) ein neues Prädikat (das Definiendum) einzuführen. Die Grundprädikate wären hier die R-Relationen, bislang definierte neue Prädikate u.a. M-, A- und N-Relationen. Die Funktion von Definitionen im Computermodell ist, daß auf Anfrage hin geprüft wird, ob das Definiendum, z.B. M-, A- oder N-Relationen, bezogen auf eine gegebene Datenmenge vorliegt. Ist dies der Fall, wird es der Faktenbasis hinzugefügt.

Mit Theoremen verhält es sich etwas anders. Die Behauptung des Theorems steckt implizit bereits in den Grundprädikaten und definierten Prädikaten. Wir sehen eine zweifache Funktion von Theoremen in einer Wissensbasis.

Regeln, die Theoreme verkörpern, haben *erstens* die Aufgabe, *die Behauptung des Theorems aus den vorliegenden Daten zu extrahieren*. Das so generierte Theorem kann ebenfalls der Faktenbasis hinzugefügt werden. In manchen Fällen ist eine Übertragung der Theoreme in das Modell aber überflüssig. Um es an einem trivialen Beispiel zu verdeutlichen: Liegt ein T-Graph vor und ist z.B. `m(a,b)` und `m(b,c)`

gegegeben, so besagt Theorem 1-3, daß auch `m(a,c)` gelten muß. Man könnte also Theorem 1-3 benutzen, um es nach unserem Regelmuster in PROLOG umzusetzen und neue Fakten herzuleiten. Im Fall von Theorem 1 ist dies aber belanglos, da es zur Herleitung von M-, A- und N-Relationen bereits Regeln *gibt*. Wurden diese Relationen vollständig abgeleitet, so sind Regeln für Theorem 1 überflüssig. Anders sieht es mit Theorem 2 aus, das uns im folgenden die Cliquenermittlung in T-Graphen möglich macht.

Die *zweite* Aufgabe von Theoremen kann die *Überprüfung der Faktenbasis* sein. Im obengenannten Beispiel muß nach vollständiger Ableitung der M-Relationen `m(a,c)` gelten. Man könnte entsprechend die Regeln für Theoreme so abwandeln, daß sie nicht ableiten, sondern nur *überprüfen*. Im Fall von Theorem 1 kann jede Implikation aber auch mit dem `leite_ab`- bzw. `zeige`-Prädikat überprüft werden, so daß keine eigenen Regeln nötig sind. Eine Repräsentation von Theorem 1 ist deshalb hier überflüssig.

Antwortet der PROLOG-Interpreter also auf

```
leite_ab(m(b,a))
```

mit `yes`, dann muß auch

```
leite_ab(m(a,b))
```

gültig sein.

Ist andererseits

```
leite_ab(a(c,d))
```

wahr, dann muß der Beweis von

```
leite_ab(a(d,c))
```

fehlschlagen.

Die nächsten Definitionen und Theoreme behandeln M-Cliquen. Die Theorie macht nur Aussagen über Cliquen, falls der HL-Graph ein T-Graph ist, d.h. der Cliquenbegriff ist gebunden an das Vorliegen eines T-Graphen. Es ist aber natürlich sinnvoll, von Cliquen zu sprechen, auch wenn kein T-Graph vorliegt.

Wir unterscheiden im folgenden deshalb zwei Cliquenbegriffe: M-Cliquen in T-Graphen und M-Cliquen in Nicht-T-Graphen. Zur Abkürzung vereinbaren wir folgende Konvention: Wenn wir von Cliquen in einem T-Graphen sprechen, benutzen wir in Anlehnung an Holland/Leinhardt den Ausdruck "M-Cliquen". Sprechen wir hingegen von Cliquen im allgemeinen Sinn, bei denen ⟨X,R⟩ kein T-Graph sein muß, benutzen wir den Ausdruck "Cliquen".

Es gilt: Bei einem gegebenen Graph ⟨X,R⟩ sind alle M-Cliquen auch Cliquen, das umgekehrte ist nicht der Fall.

Zur Repräsentation von M-Cliquen bzw. Cliquen bieten sich Listen an. Da wir für jede Relation einen eigenen Eintrag in der Faktenbasis haben, wollen wir dies analog bei Cliquen beibehalten. Wir werden also die Menge der Cliquen nicht in einem einzigen Prädikat festhalten, wie z.B. in

```
m_clique([[a,b,c],[d,e],[h,i,j,k],[f,g]]),
```

sondern für jede Clique einen eigenen Eintrag vornehmen.

Bevor Aussagen über M-Cliquen gemacht werden, müssen diese alle gefunden werden. Wir benutzen hierzu die Eigenschaft, daß innerhalb von M-Cliquen alle Knoten durch M-Kanten verbunden sind (Theorem 2-3). Eine M-Clique findet man durch das Sammeln aller M-Wähler eines beliebigen Knotenelements. Da in T-Graphen ein Knotenelement entweder zu *genau einer oder keiner* M-Clique gehört, genügt es, für jedes Element alle M-Wähler zu suchen und mehrfach gefundene Cliquen zu ignorieren. Wir gehen also die Knotenmenge X durch und suchen für jedes $x \in X$ die Menge aller Knoten, die mit x über eine M-Kante verbunden sind, genauer: Suche für alle $x \in X$ die Menge der Knoten $Y \subseteq X$, so daß für alle $y \in Y$ gilt: xMy. Die Menge $\{x\} \cup Y$ ist dann die gesuchte M-Clique. Bereits vorhandene M-Cliquen-Mengen werden ignoriert. In Abb. 9.9 (S.232) sind beispielsweise {h,i,j} alle vollständigen M-Wähler von k, so daß {k,h,i,j} eine M-Clique bildet.

Man beachte, daß dieses Suchkriterium nur für M-Cliquen gilt. In Nicht-T-Graphen kann es vorkommen, daß ein Knoten zu *mehreren* Cliquen gehört.

Anders als die bisher definierten Prädikate lassen sich Cliquen nicht durch einfaches Zurückführen auf Grundprädikate herleiten. Zwar kann die folgende M-Cliquen-Regel noch deklarativ gelesen und interpretiert werden, spätestens bei der allgemeinen Cliquensuch-Regel unten müssen wir aber die rein deklarative Interpretation von PROLOG-Prädikaten verlassen und prozedural-algorithmisch denken.

Das Auffinden einer M-Clique kann in folgenden Regelschritten erfolgen:

`[X1|L]` ist eine M-Clique, wenn

- der Graph ein T-Graph ist und
- `X` die Knotenliste ist und
- `X1` Element von `X` ist und
- `L` die Liste der M-Wahlen von `X1` ist und
- `L` nicht leer ist (Ist `L` leer, gibt es zu `X1` keine M-Wähler: instantiiere ein neues Element `X1` aus `X`, d.h. Backtracking).

```
regel(8,cliquen,m_clique([X1|L]),[m(x,y),
    ' fuer alle x y aus ', [X1|L]])
    :-
    leite_ab(graph(t_graph,true)),    % Graph ist ein T-Graph
    knoten(X),                        % X ist die Knotenmenge
    member(X1,X),                     % X1 ist Element von X
    findall(Y,leite_ab(m(X1,Y)),L),   % Sammle i.L alle Y,z.d.X1 in M-Rel.steht
    not L=[].                         % leere Liste L? -> es gibt keine
                                      % M-Waehler von X1
```

Regel 8 prüft zunächst den Graphen auf Transitivität. Liegt kein T-Graph vor, scheitert die Regel und es läßt sich keine M-Clique ableiten. Andernfalls wird die Knotenmenge mit einem Element `X1` instantiiert und mit dem Built-in-Prädikat `findall/3` alle Knoten, zu denen `X1` eine M-Relation unterhält, in eine Liste `L` gesammelt. Ist die Liste leer, gibt es keine M-Wahlen von `X1` (und keine M-Wähler von `X1`) und die `not`-Clause scheitert. Im Fall des Scheiterns erfolgt ein Backtracking zum `member`-Prädikat (`findall/3` ist nur einmal erfüllbar), so daß `X1` mit dem nächsten Knotenelement von `X` instantiiert wird. Andernfalls wird das Prädikat `m_clique/1` mit

der Liste L der M-Wahl-Knoten - erweitert um X1 - als Argument der Faktenbasis hinzugefügt.

Mit dieser Regel kann die gleiche Clique mehrmals abgeleitet werden. Da die Liste [a,b,c] in PROLOG nicht identisch ist mit [b,a,c], würden insbesondere verschiedene Permutationen der gleichen Cliquenmenge erzeugt - was natürlich unerwünscht ist. Die Permutation gleicher Cliquenmengen läßt sich leicht verhindern, wenn ein weiteres Prädikat im Körper von Regel 8 hinzugefügt wird. Das zusätzliche Prädikat müßte alle bislang gefundenen Cliquen in eine Liste L2 sammeln und jedes Element von L2 auf gleiche Menge mit dem aktuellen Cliquenkandidaten [X1|L] prüfen. Aus zwei Gründen erscheint diese Lösung unschön. Erstens haben wir im Regelinterpreter bereits einen Mechanismus, der verhindert, daß bereits abgeleitete Fakten nochmals hergeleitet werden. Zweitens taucht das gleiche Problem der Ignorierung von Listenpermutationen weiter unten noch öfter auf. Wir implementieren diese Funktion deshalb auf Metaebene und erweitern und ändern den Regelinterpreter wie folgt.

```
leite_ab(X) :-
   check(X),
   faktum(X,_).

leite_ab(X) :-
   regel(Nr,_,X,Ursach),
   not schon_vorhanden(X),
   schreibe_liste([
   'Mit Regel ',Nr,' abgeleitet: ',X,
   '   -> Faktenbasis'
                  ]),
   asserta(faktum(X,[Nr|Ursach])).
```

Eingefügt wurde in der zweiten leite_ab-Clause not schon_vorhanden(X), welches vor dem Hinzufügen von X zur Faktenbasis prüft, ob dieses schon vorhanden ist. Hierbei lassen sich dann bei Prädikaten, die Listen enthalten, Permutation leicht zurückweisen.

```
schon_vorhanden(m_clique(X)) :-
   findall(Z,faktum(m_clique(Z),_),L),
   test_auf_gleiche_menge(X,L).

schon_vorhanden(X) :-
   faktum(X,_).

test_auf_gleiche_menge(_,[]) :- !,fail.
test_auf_gleiche_menge(X,[K|Rest]) :-
   gleiche_menge(X,K),!.
test_auf_gleiche_menge(X,[_|Rest]) :-
   test_auf_gleiche_menge(X,Rest).
```

Analog wie die anderen Regelprädikate findet Regel 8 bei *einer* Anfrage genau *eine* M-Clique. *Alle* M-Cliquen werden generiert durch

```
leite_ab(m_clique(X)).
```

mit einem anschließenden Fail, also einem vom Benutzer angestoßenen Backtracking.

Theorem 3 bringt keine neuen Informationen, die nicht schon mit bestehenden Regeln ableitbar wären, so daß auf eine PROLOG-Codierung verzichtet wird.

In Definition 4 wird die Ordnungsrelation A* auf M-Cliquen definiert. Die entsprechende PROLOG-Regel läßt sich unter Verwendung eines Hilfsprädikats wie folgt schreiben.

```
regel(9,cliquen,a_stern(U,V),
   [a(x,y),' fuer alle x aus ', U,' und y aus ',V])
   :-
   leite_ab(m_clique(U)),
   leite_ab(m_clique(V)),
   ungleich(U,V),
   sind_alle_a_waehler_von(U,V).

sind_alle_a_waehler_von([],_).
sind_alle_a_waehler_von([X|Rest],V) :-
   ist_ein_a_waehler_von(X,V),
   sind_alle_a_waehler_von(Rest,V).

ist_ein_a_waehler_von(_,[]).
ist_ein_a_waehler_von(X,[Y|Rest]) :-
   leite_ab(a(X,Y)),
   ist_ein_a_waehler_von(X,Rest).
```

Regel 9 leitet zwei M-Cliquen U und V ab, die verschieden sein müssen und prüft mit dem Prädikat `sind_alle_a_waehler_von(U,V)`, ob von allen Mitgliedern der U-Clique zu allen Mitgliedern der V-Clique eine A-Relation vorliegt. Ist dies nicht der Fall und wurden `U` und `V` instantiiert aufgerufen, scheitert die Regel. Wurden `U` und `V` uninstantiiert aufgerufen, erfolgt ein Backtracking zu `leite_ab(m_clique(V))` bzw. `leite_ab(m_clique(U))`.

Alle A*-Relationen werden wieder erzeugt mit

```
leite_ab(a_stern(X,Y)),fail.
```

9.4.2 Triadentypen und spezielle T-Graphen

Die 16 in Abb. 9.4 (S.188) aufgelisteten Triadentypen können mit Regeln der folgenden Art repräsentiert werden.

```
regel(10,triadentyp,triadentyp([X,Y,Z],'201'),
   [m(X,Z),m(X,Y),n(Y,Z)])
   :-
   leite_ab(m(X,Z)),
   leite_ab(m(X,Y)),
   leite_ab(n(Y,Z)).
```

Die Regel besagt, daß die Triade [X,Y,Z] vom Triadentyp 201 ist, wenn m(X,Z), m(X,Y) und n(Y,Z) gelten. Jedem Triadentyp aus Abb. 9.4 entspricht hierbei eine Regel.

Aufgrund der Tatsache, daß eine Triade sechs geordnete Knotenpaare umfaßt, die jeweils durch R verbunden sind oder nicht, ist jede Triade in einem von $2^6 = 64$ Zu-

ständen. Abb. 9.4 gibt nur die 16 ihrer *Struktur* nach unterscheidbaren Triadentypen wieder. Während die Triaden 003 und 300 beispielsweise völlig determiniert sind, gilt dies nicht für die Triade 012. Deren Struktur liegt vor, wenn in der Triade [a,b,c] eine der folgenden sechs Möglichkeiten realisiert ist.

```
a(a,b), n(a,c), n(c,b)
a(b,a), n(a,c), n(c,b)
a(a,c), n(a,b), n(c,b)
a(c,a), n(a,b), n(c,b)
a(b,c), n(a,c), n(a,b)
a(c,b), n(a,c), n(a,b)
```

Wir bräuchten also eigentlich sechs Regeln für die 012-Struktur. Insgesamt würde dies die Anzahl der Triadentyp-Regeln beträchtlich erhöhen, nämlich von 16 auf 64. Man kann das Problem umgehen, wenn gefordert wird, daß die Argumente beim Regelaufruf uninstantiiert sind. Der PROLOG-Interpreter sucht sich dann selbst die zu jeder Struktur passenden Instantiierungen.

Für jeden Triadentyp kann festgestellt werden, ob dieser transitiv, intransitiv oder leer transitiv ist. Eine entsprechende Regel für intransitive Triaden könnte lauten:

```
regel(x, transtest, transtyp(X,intransitiv),...)
   :-
   leite_ab(triadentyp(X,Y),
   member(Y,['021c','030c', '111d','111u','120c','201','210']).
```

Analog müßten zwei weitere Regeln für transitive und leer-transitive Triaden eingeführt werden.
Ein Nachteil dieser Darstellung ist, daß bei vollständiger Herleitung aller Tripel die Faktenbasis unnötig stark erweitert wird. Bei n Knoten gibt es

$$\frac{n\cdot(n-1)\cdot(n-2)}{6}$$

Triaden, bei zehn Knoten also bereits 120 Triaden. In diesem Fall würden also weitere 120 Fakteneinträge hinzugefügt werden, obwohl Fakten zu den einzelnen Triaden (Triadentyp) bereits existieren. Wir nehmen deshalb die Information zur Transitivität in das Triadentyp-Prädikat in einer dritten Argumentstelle auf, so daß Regel 10 wie folgt verändert wird:

```
regel(10,triadentyp,triadentyp([X,Y,Z],'201',intransitiv),
    [m(X,Z),m(X,Y),n(Y,Z)])
   :-
   leite_ab(m(X,Z)),
   leite_ab(m(X,Y)),
   leite_ab(n(Y,Z)).
```

Die Regeln für die restlichen 15 Triadentypen können analog dargestellt und dem vollständigen Programmcode des Anhangs entnommen werden.

Die nächsten Regeln behandeln Spezialfälle von T-Graphen. Die Regeln 26-33 prüfen Restriktionen der Kanten, wie sie in Tab. 9.1 (S.186) zusammengefaßt sind und leiten die entsprechenden Graphentypen ab.

Zunächst werden Regeln eingeführt für Graphen, in denen genau eine der Relation M, A oder N präsent ist. Regel 26 leitet den vollständig verbundenen Graphen ab ("Einmütigkeit" im Sinn von Cartwright/Harary, d.h. Transitivität, $M \neq \emptyset$, N, $A = \emptyset$), Regel 27 Hierarchie im strengen Sinn ("transitive tournament", d.h. Transitivität, $A \neq \emptyset$, $M,N = \emptyset$) und Regel 28 den völlig unverbundenen Graphen (Transitivität, $N \neq \emptyset$, $M,A = \emptyset$).

```
regel(26,graph,graph(completely_connected,true),
   [graph(t_graph,true),'es gibt m(x,y): ',m(X,Y),
   ' es gibt kein n(x,y)',' es gibt kein a(x,y)'])
   :-
   leite_ab(graph(t_graph,true)),
   leite_ab(m(X,Y)),
   not leite_ab(n(_,_)),
   not leite_ab(a(_,_)).

regel(27,graph,graph(transitive_tournament,true),
   [graph(t_graph,true),' es gibt a(x,y): ',a(X,Y),
   ' es gibt kein m(x,y)',' es gibt kein n(x,y)'])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(m(_,_)),
   not leite_ab(n(_,_)),
   leite_ab(a(X,Y)).

regel(28,graph,graph(completely_disconnected,true),
   [graph(t_graph,true),' es gibt ein n(x,y): ',n(X,Y),
   ' es gibt kein m(x,y)',' es gibt kein a(x,y)'])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(m(_,_)),
   not leite_ab(a(_,_)),
   leite_ab(n(X,Y)).
```

Die nächste Regelgruppe behandelt die Fälle, in denen genau zwei Relationen vorkommen. Ein transitiver Graph bildet eine Quasi-Serie, wenn M- und A-, aber keine N-Relation vorhanden ist.

```
regel(29,graph,graph(quasi_series,true),
   [graph(t_graph,true),' es gibt ein m(x,y): ',m(X1,X2),
   ' es gibt kein n(x,y)',' es gibt ein a(x,y): ',a(Y1,Y2)])
   :-
   leite_ab(graph(t_graph,true)),
   leite_ab(m(X1,X2)),
   not leite_ab(n(_,_)),
   leite_ab(a(Y1,Y2)).
```

Ein transitiver Graph bildet eine partielle Ordnung (von Personen), wenn A- und N-, aber keine M-Relationen vorhanden sind.

```
regel(30,graph,graph(partial_order,true),
   [graph(t_graph,true),' es gibt kein m(x,y)',
   ' es gibt ein n(x,y): ',n(X1,X2),
   ' es gibt ein a(x,y): ',a(Y1,Y2)])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(m(_,_)),
   leite_ab(n(X1,X2)),
   leite_ab(a(Y1,Y2)).
```

Ein Graph mit fehlender A-Kante, aber gegebenen M- und N-Relationen leitet schließlich über zu den Spezialfällen historisch älterer Balancemodelle. Diese speziellen Modelle lassen sich immer als T-Graphen plus zusätzliche Bedingungen charakterisieren.
Fehlt im transitiven Graphen die A-Relation, liegt das Modell des gruppierbaren Graphen von Davis (1967) vor.

```
regel(31,graph(clusterable_graph,true),
   [graph(t_graph,true),'es gibt m(x,y): ',
   m(X1,X2),' es gibt n(x,y): ',
   n(Y1,Y2),' es gibt kein a(x,y)'])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(a(_,_)),
   leite_ab(m(X1,X2)),
   leite_ab(n(Y1,Y2)).
```

Die Gruppen können mit der Regel für M-Cliquen hergeleitet werden. Zwischen den Cliquen existieren dann keine R-Relationen.

Alternativ könnte der Clusterable Graph auch über Restriktion von Triadentypen hergeleitet werden. Da der Graph keine intransitiven Triaden und keine A-Relationen enthalten darf, ist ein Clusterable Graph dadurch charakterisiert, daß nur die Triadentypen 003, 102 und 300 vorkommen dürfen.

Die anderen beiden "klassischen" Balancemodelle - Structural Balance und Ranked Clusters - erhält man ausschließlich durch Beschränkung der Triadentypen.
Strukturelle Balance im Sinn von Cartwright/Harary (1956) liegt vor, wenn der Graph transitiv ist und einzig Triaden des Typs 102 und 300 auftreten.

```
regel(32,graph,graph(structural_balance,true),[graph(t_graph,true),
   'es gibt nur Triaden 102, 300',
   'es gibt keine Triaden 003, 012, 021u, 021d, 030t, 120u, 120d'])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(triadentyp(_,'003',_)),
   not leite_ab(triadentyp(_,'012',_)),
   not leite_ab(triadentyp(_,'021u',_)),
   not leite_ab(triadentyp(_,'021d',_)),
   not leite_ab(triadentyp(_,'030t',_)),
   not leite_ab(triadentyp(_,'120u',_)),
   not leite_ab(triadentyp(_,'120d',_)).
```

Regel 32 verifiziert, daß ein T-Graph vorliegt, und läßt nur die Triadentypen 102 und 300 zu, indem sie fordert, daß die anderen transitiven und leer-transitiven Triaden nicht vorkommen dürfen.
Das letzte spezielle Modell schließlich ist das Ranked-Clusters-Modell von Davis/Leinhardt (1972). Dieses ist gegeben bei Transitivität und Fehlen von 012-Triaden.

```
regel(33,graph,graph(ranked_clusters,true),
   [graph(t_graph,true), 'es gibt keine 012-Triaden'])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(triadentyp(_,'012',_)).
```

9.4.3 Prädikate zu Nicht-T-Graphen

Wir müssen nun einige PROLOG-Prädikate für Nicht-T-Graphen einführen. Der Grund ist, daß vollständige Transitivität empirisch betrachtet der Ausnahmefall ist, reale Strukturen in der Regel intransitiv sind und das inhaltliche Axiom von HLT eine Tendenz von intransitiven Graphen zu weniger intransitiven Graphen postuliert. Die zentrale Definition ist dabei die der Clique in einem Nicht-T-Graphen (M-Cliquen in Nicht-T-Graphen bezeichnen wir - wie oben bereits gesagt - einfach als "Cliquen").

9.4.3.1 Ermittlung von Nicht-T-Graph-Cliquen und NP-Vollständigkeit

Die Ermittlung von Cliquen in Nicht-T-Graphen ist aus zweierlei Gründen angebracht: Erstens ist es informativ, auch in nicht transitiven Graphen alle Cliquen errechnen zu können. Wichtiger in unserem Kontext aber ist zweitens, daß wir unten verschiedene Hypothesen formulieren werden, wie ein intransitiver unbalancierter Graph in einen transitiven balancierten überführt wird. Dabei gehen wir von Cliquen als Initialisierungsträgern aus. Folglich müssen wir die Cliquen in einem Nicht-T-Graphen zuerst einmal ermitteln.

Das Auffinden von Cliquen in einem T-Graphen war - auf der Basis von Theorem 2 - trivial. Für einen beliebigen Knoten x bildeten alle Knoten, die mit x über M-Kanten verbunden waren, eine M-Clique, und ein Knoten gehörte entweder zu keiner oder genau einer M-Clique. Im Gegensatz dazu kann ein Knoten in einem Nicht-T-Graphen zu *mehreren* Cliquen gehören, so daß die Cliquendefinition nach Theorem 2 und die obige Regel hier nicht anwendbar sind. Beispielsweise gehört im nicht-transitiven Graphen von Abb. 9.6 der Knoten d sowohl zu Clique {a,d} als auch zu {b,c,d}.

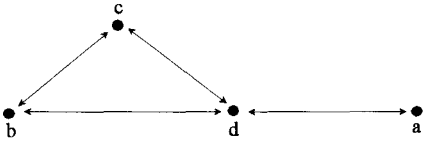


Abb. 9.6: Beispiel für zwei Nicht-T-Graph Cliquen mit Liaisonknoten d

Knoten d in Abb. 9.6 wird als Liaisonperson bezeichnet. Eine *Liaisonperson* ist allgemein eine Person, die zwei Cliquen gemeinsam haben, welche sich sonst nicht überlappen. Gehören mehrere Knoten zwei überlappenden Cliquen an, so spricht man von *cocliqualen* Knoten (Kappelhoff 1987: 46).

Wir stützen uns bei dem Cliquensuch-Algorithmus auf die allgemeine, graphentheoretische Cliquendefinition. Diese Definition ist die einfachste und zugleich restriktivste Bestimmung, sie entspricht aber völlig dem Konzept der M-Clique in einem transitiven Graphen. In dieser Definition, die auf Luce/Perry (1949) zurückgeht, werden Cliquen als "maximal vollständige Sub-Graphen" definiert.[69]

(Cliquen-)Definition (Luce/Perry 1949, vgl. auch Harary 1969: 20):
Ein Sub-Graph $\langle X',R' \rangle$ eines Graphen $\langle X,R \rangle$ ist eine Clique (im Sinn eines maximal vollständigen Sub-Graphen) genau dann, wenn
(1) alle $x \in X'$ durch M-Relationen verbunden sind (Vollständigkeit) und
(2) kein weiterer Knoten hinzugefügt werden kann, ohne daß (1) verletzt wird (*maximale* Vollständigkeit).

Ein Überblick über andere, schwächere Cliquendefinitionen findet sich in Kappelhoff (1987). In unserem Zusammenhang sind diese Definitionen mit einer - weiter unten diskutierten - Ausnahme aber nicht weiter wichtig.

Ein korrektes und vollständiges Cliquenermittlungs-Verfahren im Sinn der Cliquendefinition von Luce/Perry (1949) findet also alle und nur die maximal vollständigen Teilgraphen. Da es bei einer größeren Datenmenge unmöglich ist, die Cliquen durch optische Inspektion zu ermitteln, wurde die Suche nach solchen Algorithmen bereits in den fünfziger Jahren mit dem Aufkommen der ersten Rechner als Notwendigkeit erachtet.

Das Auffinden von Cliquen in einem intransitiven Graphen ist nicht trivial und wird als "Cliquenproblem" unglücklicherweise sogar zur Klasse der NP-Rechenprobleme gezählt (Reischuk 1990: 242-243, Claus 1986: 97-98). Probleme dieses Typs treten insbesondere dort auf, wo die Lösung aus dem Durchsuchen einer großen Anzahl von Fällen besteht und ein vollständiges Resultat nur durch systematisches Ausprobieren aller Möglichkeiten gegeben ist. Mit NP-Rechenproblemen hat es genauer folgende Bewandtnis (vgl. zur Darstellung der NP-Problematik Goldschlager/Lister 1990: 94-105).

Wir haben eingangs gesagt, daß es für bestimmte Probleme eine algorithmische Lösung geben kann, diese aber so viele Betriebsmittel in Form von Speicher und Zeit verbrauchen kann, daß sie praktisch nicht verwendbar ist. Die Komplexitätstheorie behandelt genau diese Probleme und drückt den Betriebsmittelbedarf eines Algorithmus als Funktion der Anzahl der Eingaben n (hier: Knoten) aus. Ein Algorithmus, dessen Ausführungszeit proportional zur Anzahl n der Eingabedaten ist, zeigt ein

[69] Die Begriffe "Sub-Graph" und "Vollständigkeit eines Graphen" wurde in dem graphentheoretischen Teil bereits eingeführt.

vernünftiges Leistungsverhalten, da bei z.B. doppelter Anzahl von Eingaben der Algorithmus doppelt so viel Zeit braucht.

Generell gibt es zwei Typen von Algorithmen, wovon der erste ein völlig unakzeptables Zeitverhalten hat. Algorithmen, deren (asymptotisches) Verhalten 2^n oder allgemein c^n (mit einer Konstanten c) ist, heißen *exponentielle Algorithmen*. Ist die Ausführungszeit bei n = 10 Eingaben also 0.001 Sekunden, so ist sie bei n=100 schon bei 10^{14} Sekunden, das sind Jahrhunderte. Außer bei kleinsten Eingabedaten sind exponentielle Algorithmen normalerweise undurchführbar. Der zweite Typ von Algorithmen zeigt ein akzeptableres Verhalten: Algorithmen, deren asymptotisches Verhalten n, n^2 oder allgemein n^c (für eine Konstante c) ist, heißen *polynomiale Algorithmen*. Polynomiale Algorithmen tendieren dazu, für eine vernünftige Zahl von Eingabedaten ausführbar zu sein. In erster Annäherung kann man sagen, daß Algorithmen mit polynomialem Zeitverbrauch akzeptabel sind und alle anderen nicht.[70] Da eine Theorie über durchführbare Algorithmen weitgehend maschinenunabhängig formuliert werden kann, ist die allgemeine Frage, *welche* Aufgaben in polynomialem Zeitverbrauch berechnet werden können. Für zahlreiche Aufgabenstellungen ist der Beweis erbracht worden, daß sie in polynomialem Zeitverbrauch berechnet werden können, für einige ist gezeigt worden, daß sie dies nicht tun und für viele Aufgaben ist die Frage nach einem schnellen (polynomialen) Algorithmus offen. Das Cliquenproblem gehört nun ebenso wie das bekanntere Problem des Handlungsreisenden (Travelling Salesman)[71], das Stundenplanproblem und das Finden eines Hamilton-Zyklus genau zu dieser Klasse von Fragestellungen, für die weder ein polynomialer Algorithmus gefunden wurde, noch ein Beweis existiert, daß es einen solchen nicht gibt.

Die eben genannten Probleme werden zur Klasse der NP-Rechenprobleme gezählt.[72] Diese Probleme haben folgende *gemeinsame Eigenschaft*: Ist einmal eine Lösung für einen Beispielfall gelungen, so ist es leicht zu *verifizieren*, daß die Lösung korrekt ist. In diesem Fall gibt es nämlich einen (Meta-)Algorithmus für den Beispielfall, der für die vorgeschlagene Lösung in *polynomialer* Laufzeit bestimmt, ob die Lösung korrekt ist oder nicht. Darüberhinaus wurde 1971 von Cook eine wichtige Entdeckung gemacht: *Falls ein Algorithmus mit polynomialem Zeitverhalten jemals für eines der Probleme gefunden wird, dann gibt es einen Algorithmus für jede Aufgabenstellung in der Menge NP*. Alle Aufgabenstellungen, die zu diesen schwersten Aufgabenstellungen gehören, heißen NP-vollständig. Es läßt sich nun zeigen, daß das Cliquenproblem NP-vollständig ist (Beweis in Reischuk 1990: 242-243).

---

[70] Natürlich gibt es auch polynomiale Algorithmen, die nicht durchführbar sind, z.B. einer, der $5n^{1000}$ Stunden benötigt. Exponentielle Algorithmen tendieren aber dazu, selbst für eine geringe Anzahl von Eingabedaten den Betriebsmittelbestand zu überschreiten. Aber auch hier gibt es Ausnahmen (Goldschlager/Lister 1990:101).

[71] Das Travelling Salesman Problem besteht darin, ob es bei einer Straßenkarte mit n Städten einem Handlungsreisenden bei gegebener Kilometerbeschränkung möglich ist, jede Stadt genau einmal bei einer Rundreise zu besuchen.

[72] Der Name "NP" ist von einem präziseren mathematischen Mengenbegriff hergeleitet (Goldschlager/Lister 1990: 107).

Unter Berechenbarkeitsaspekten sind damit alle NP-vollständigen Probleme äquivalent. Ein polynomialer Algorithmus für das Travelling Salesman Problem würde einen ebensolchen für das Cliquenproblem und alle anderen NP-Probleme ergeben. Da bislang alle Versuche von Experten, polynomiale Algorithmen für NP-vollständige Aufgabenstellungen in ihrem Sachgebiet zu finden, fehlschlugen, wird angenommen, daß NP-vollständige Probleme undurchführbar sind (allerdings haben auch zahlreiche Experten versucht, die Undurchführbarkeit NP-vollständiger Aufgabenstellungen zu zeigen und sind bislang ebenso gescheitert).

Werden nun Computerlösungen für NP-vollständige Probleme gebraucht, so müssen die Anforderungen an die Lösung auf einen der folgenden Punkte abgeschwächt werden (Goldschlager/Lister 1990: 105):

- Eine erste Möglichkeit besteht darin, nach einer Näherungslösung zu suchen (die besser ist, als überhaupt keine Lösung). Statt etwa beim Handelsreisenden-Problem die *kürzeste* Rundreise in unvertretbar viel Rechenzeit zu bestimmen, begnügt man sich mit einer *vertretbar kurzen* Rundreise.
- Eine zweite Möglichkeit ist, einen Algorithmus zu finden, der bei durchschnittlichen Eingabemengen gut arbeitet und nur in den schlechtesten Fällen exponentielles Verhalten zeigt. Es besteht dann die Hoffnung, daß der Algorithmus bei den meisten Inputs, die er in der Praxis erhält, in vertretbarer Zeit endet.
- Eine dritte Möglichkeit liegt schließlich darin, die Bedingung, daß der Algorithmus korrekt arbeitet, aufzugeben und Fehler in Kauf zu nehmen. Eine weitere, von Goldschlager/Lister nicht genannte Möglichkeit ist, die Forderung nach Vollständigkeit aufzugeben.

Die algorithmische Lösung des Cliquenproblems ist also mit einigen Schwierigkeiten, vor allem bei Netzen mit größerer Anzahl von Knoten verbunden. Angesichts der NP-Problematik erscheint die Forderung von Collanis (1987: 131) an einen Algorithmus zur Lösung des Cliquenproblems als nicht ganz einsichtig. von Collani stellt folgende Anforderungen an einen Cliquensuch-Algorithmus auf:

1. Der Algorithmus muß korrekt und vollständig sein, d.h. er darf keine Clique liefern, die nicht vorkommt und es müssen alle Cliquen geliefert werden;
2. Die Lösung muß mit vertretbarem Speicher- und Programmieraufwand gefunden werden;
3. Die Lösung muß in möglichst geringer Rechenzeit produzierbar sein.

Die dritte Forderung kollidiert dabei mit der ersten Forderung. Denn angenommen, man verwendet zur Minimierung der Rechenzeit Heuristiken und Näherungsalgorithmen, dann ist die Gefahr groß, daß der Algorithmus nicht vollständig oder nicht korrekt ist. Oder man legt mehr Wert auf Vollständigkeit und Korrektheit, dann ist der Algorithmus für sehr große Knotenmengen praktisch nicht mehr verwendbar.

Bislang existiert eine Vielzahl unterschiedlich komplexer und praktisch brauchbarer Algorithmen, die sowohl in der soziometrischen als auch in der computerwissenschaftlichen Fachliteratur diskutiert werden. Ein Verfahren, das vollständig und korrekt ist, wurde erstmals von Harary/Ross (1957) vorgestellt. Der Harary/Ross-Algorithmus basiert auf einer 0-1-Matrixdarstellung des Graphen und geht grob gesagt wie folgt vor: Zunächst werden Knotenpaare betrachtet, dann Knotentripel

usw., bis eine maximal vollständige Teilmenge gefunden ist. Die so gefundene Menge bildet die erste Clique und das Verfahren wird für ein anderes Knotenpaar wiederholt. Jeder Versuch, eine neue Clique zu finden, erfordert, daß alle Punktepaare, zwischen denen Kanten bestehen, untersucht werden. Die Methode von Harary/Ross wird sehr langsam, wenn die Anzahl der Knoten, insbesondere aber die der Kanten, steigt. Andererseits liefert dieses Vorgehen vollständige und korrekte Lösungen. Ein weiterer, ebenfalls vollständiger und korrekter Algorithmus wäre, alle Teilmengen X' der Größe k danach zu prüfen, ob sie eine k-Clique bilden. Da es 2^n Teilmengen gibt, würde dieses Verfahren 2^n Schritte zur Lösung benötigen. Bei anderen Algorithmen, die geschickte und zeitoptimierende Heuristiken benutzen, zeigte sich, daß diese z.T. nicht vollständig sind. Dies trifft etwa zu auf den Algorithmus von Rattinger (1973). Wie Freeman/Libhart (1986) gezeigt haben, identifiziert Rattingers Methode - obgleich elegant und ziemlich schnell - unter bestimmten Bedingungen nicht alle maximal vollständigen Teilgraphen. Weitere Algorithmen werden in Knoke/Kuklinski (1982) vorgestellt. Eine zusammenfassende und kritische Diskussion findet sich bei von Collani (1987). Von Collani empfiehlt die Algorithmen von Bron/Kerbosch (1973) oder Knödel (1968), die aber nur unter bestimmten, nicht immer erfüllten Voraussetzungen anwendbar sind. Sind die Bedingungen für die Anwendung nicht gegeben, empfiehlt von Collani das Verfahren von Bierstone in der Fassung von Mulligan/Corneil (1972), das wiederum den Nachteil hat, daß der Speicherbedarf von der Anzahl der ermittelten Cliquen abhängt. Aus diesem Literaturüberblick wird deutlich, daß das Aufsuchen von Cliquen in der Praxis große Probleme bereitet.

Alle üblichen Cliquensuch-Algorithmen basieren in der Regel auf Matrizendarstellung der Graphen und nicht auf symbolischen Repräsentationen. Wir entwickeln nun ein Berechnungsverfahren für Cliquen in symbolisch codierten Graphen *auf der Basis von Zyklen* (zum Zyklusbegriff vgl. Kap. 8.1.2, S.168f.), das unseres Wissens nicht realisiert ist. Ausgehend von der Cliquendefinition als maximal vollständiger Sub-Graph suchen wir einen vollständig M-verbundenen Teil-Graphen $X' \subseteq X$, so daß es keinen Knoten außerhalb von X' gibt, der zu allen Mitgliedern von X' M-verbunden ist. Wie anhand der beiden Cliquen {a,b,c,d,e} und {d,e,f} von Abb. 9.7 leicht zu ersehen ist, gehören alle Knoten innerhalb einer Clique einem M-Zyklus Z an, der maximal ist in dem Sinn, daß es keinen Knoten außerhalb von Z gibt, der zu allen Zyklusknoten Z M-verknüpft ist. Der grundlegende Ansatz dieses Cliquengenerierungs-Algorithmus besteht also darin, genau jenen Zyklus zu finden, der Bedingung (2) der Cliquendefinition erfüllt. Wir hoffen dabei, daß der Algorithmus im Sinn der zweiten von Goldschlager/Lister genannten Möglichkeit bei "durchschnittlichen Eingabemengen" gut arbeitet. Optimierungsvarianten kann man sich nachträglich noch überlegen.

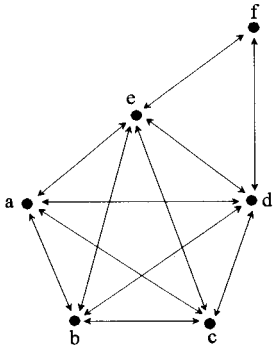


Abb.9.7: Beispielgraph zur Entwicklung des zyklenbasierten Cliquensuch-Verfahrens

Verdeutlichen wir das Problem des Auffindens maximal vollständiger Teilgraphen mit Zyklen am Beispiel des Graphen in Abb. 9.7. In Abb. 9.7 gibt es viele M-Zyklen, z.B.
⟨a,e,b,a⟩
⟨a,e,a⟩
⟨a,e,f,d,c,a⟩.
Eine Teilmenge der möglichen Zyklen beinhaltet genau die gesuchten Cliquen, nämlich {a,e,d,c,b,a} und {e,f,d,e}. Unser Cliquenfinde-Algorithmus muß also verhindern, daß

1. der Zyklus und damit die Cliquenmenge nicht zu klein ist (z.B. darf {a,e,b,a} nicht als Clique eingestuft werden, da diese Menge nur Teil einer umfassenderen Clique ist)
2. der Zyklus/die Cliquenmenge nicht zu groß wird (z.B. nicht {a,b,c,d,f,e,a} umfassen).

Der ersten Bedingung der allgemeinen Cliquendefinition genügt also derjenige Zyklus Z für den gilt, daß alle Elemente von Z untereinander M-verbunden sind. Dies sind z.B. die Zyklen (Z1) ⟨f,d,e,f⟩, (Z2) ⟨a,b,c,a⟩, (Z3) ⟨d,e,a,d⟩. Der Zyklus (Z4) ⟨f,e,a,d,f⟩ hingegen genügt der Bedingung nicht, da f und a nicht M-verknüpft sind. Von diesen Zyklen bildet nur (Z1) eine Clique, d.h. die zweite Bedingung muß alle anderen Zyklen als Cliquenkandidaten verwerfen.

Die zweite Bedingung lautet, daß es kein Element x außerhalb des Zyklus Z geben darf, das zu *allen* $z \in Z$ in M-Relation steht. Diese Bedingung filtert z.B. (Z2) aus der Menge der gültigen Cliquen aus, da es Elemente gibt, die sowohl zu a als auch b und c in M-Relation stehen (nämlich die Knoten e und d). Das Analoge gilt für (Z3). Der Zyklus (Z1) ⟨f,d,e,f⟩ hingegen genügt dem Kriterum, da zwar d und e zu allen nicht Z-Knoten in M-Relation stehen, nicht jedoch f. Bestünde zwischen f und a eine M-Relation, so würde (Z1) nicht als Clique eingestuft, da es dann ein Element a geben würde, das zu d, f und e in M-Relation stünde.

Der Algorithmus lautet somit genauer wie folgt: Sei X die Knotenmenge eines Graphen. Finde einen M-Zyklus Z zwischen einem Knoten $x \in X$ derart, daß es kein

Element aus der Differenzmenge zwischen X und Z gibt, das zu allen $z \in Z$ M-verbunden ist und für alle $x,y \in Z$ gilt: xMy.

Wenn wir wieder von der Knotenmenge X ausgehen und jedes Element bezüglich aller seiner M-Zyklen abprüfen, ergibt sich in PROLOG-naher Schreibweise:
Z ist eine Clique wenn gilt:

- X ist die Knotenmenge und
- $x \in X$ und
- es gibt einen M-Zyklus Z von x nach x und
- es gibt kein $x \in X \setminus Z$, so daß für alle $z \in Z$ gilt: xMz und
- für alle $x,y \in Z$: xMy.

Den Kern des Verfahrens bildet das Prädikat `pfad/4`, das wir später auch noch benötigen, so daß wir den Zyklus als Spezialfall von `pfad/4` definieren. Das Prädikat `pfad(X,Y,Pfad,Rel_typ)` sucht in einem Graphen einen azyklischen Weg von `X` nach `Y`, in dem kein Knoten mehrmals auftaucht. Die durchlaufenen Knoten werden in der Liste `Pfad` gesammelt (vgl. Bratko 1987: 244-246).

```
pfad(X,Y,Pfad,Rel_typ) :-
   pfad1(X,[Y],Pfad,Rel_typ).

pfad1(X,[X|Pfad1],[X|Pfad1],Rel_typ).

pfad1(X,[Y|Pfad1],Pfad,Rel_typ) :-
   R=..[Rel_typ,Y1,Y],
   leite_ab(R),
   not member(Y1,Pfad1),
   pfad1(X,[Y1,Y|Pfad1],Pfad,Rel_typ).
```

Das Prädikat ist so allgemein formuliert, daß es Pfade von frei bestimmbarem Relationstyp suchen kann. Den Relationstyp legt das vierte Argument fest. Die Variable `Rel_typ` muß in unserem Fall mit einer der Relationen R, M, A oder N instantiiert sein, speziell im Fall der Cliquenfindung also mit M. Wird `pfad/4` mit `m` an vierter Argumentstelle aufgerufen, werden also M-Pfade in Graphen gefunden. Wenn im Kontext der Cliquenfindung von Pfaden die Rede ist, sind immer M-Pfade gemeint.
Durch Backtracking werden *alle möglichen* M-Pfade in einem Graphen entdeckt, wobei die Reihenfolge der gefundenen Pfade von der Reihenfolge der Relationen in der Faktenbasis abhängt. Für den Graphen in Abb. 9.7 liefert `pfad/4` z.B.

```
?- pfad(a,b,W,m).
W  = [a,e,f,d,b];
W  = [a,e,b];
W  = [a,d,f,e,c,b];
....
?- pfad(k,a,W,m).
no
```

Ein Zyklus ist ein spezieller Pfad, der wieder zum Ausgangsknoten zurückführt: zum Ausgangselement x gibt es einen Pfad derart, daß von x zu (dem benachbarten) Knoten y ein Pfad existiert und von y zu x ein Pfad besteht.

```
zyklus(X,Pfad,Rel_typ) :-
   R=..[Rel_typ,Y,X],
   leite_ab(R),
   pfad(X,Y,Pfad).
```

Damit ist das grundlegende Zyklus-Prädikat definiert und wir können die Cliquen-Regel nach dem entwickelten Algorithmus einführen.

```
regel(34,cliquen,clique(Z),['m(x,y) fuer alle x,y aus ', Z, ' nicht
es gibt ein z aus ', Rest, ' so dass: m(z,x) fuer alle x aus ', Z])
   :-
   knoten(X),
   member(X1,X),
   zyklus(X1,Z,m),
   sind_alle_m_verbunden(Z),
   differenz(X,Z,Rest),
   nicht_m_verbunden(Rest,Z).
```

Das Prädikat `zyklus/3` sucht - ausgehend von dem instantiierten Knoten `X1`- einen Zyklus `Z`, wobei `Z` als *potentielle* Clique betrachtet wird. Die weiteren Prädikate beweisen, daß der Zyklus `Z` eine Clique ist. `sind_alle_m_verbunden(Z)` verifiziert, daß alle Elemente aus `Z` untereinander M-verknüpft sind. `differenz(X,Z,Rest)` berechnet die Differenzmenge `Rest` zwischen der Knotenmenge `X` und der Zyklusmenge `Z` und `nicht_m_verbunden(Rest,Z)` ist wahr, wenn kein Element von `Rest` zu allen Elementen von `Z` in M-Relation steht.

Backtracking kann an folgenden Stellen erfolgen: das Prädikat `zyklus/3` schlägt fehl, wenn es ausgehend vom aktuellen Knotenelement keinen Zyklus gibt, es erfolgt ein Backtracking zum `member`-Prädikat, so daß das nächste Knotenelement `X1` instantiiert wird. Die Prädikate `sind_alle_m_verbunden/1` und `nicht_m_verbunden/2` können fehlschlagen und zum Prädikat `zyklus/3` backtracken, so daß ein alternativer Zyklus gefunden wird bzw. ein weiteres Backtracking zu `member/2` erfolgt.

Ein *einmaliger* Regelaufruf bestimmt *eine* Clique. Weitere Cliquen werden wieder durch ein benutzerangestoßenes Backtracking gefunden.
Nachzutragen bleiben noch die beiden Hilfsprädikate `nicht_m_verbunden/2` und `sind_alle_m_verbunden/1`.

```
nicht_m_verbunden([],_).
nicht_m_verbunden([X|R],Z) :-
   not ist_m_verbunden(X,Z),
   nicht_m_verbunden(R,Z).

ist_m_verbunden(_,[]).
ist_m_verbunden(X,[Y|R]) :-
   leite_ab(m(X,Y)),
   ist_m_verbunden(X,R).
```

`nicht_m_verbunden(Rest,Z)` arbeitet alle Restelemente, die nicht in `Z` enthalten sind, rekursiv ab unter Benutzung von `ist_m_verbunden`. `ist_m_verbunden/2` ist erfolgreich, wenn das Restelement `X` zu *allen* Elementen der Cliquenkandidaten-Menge `Z` in M-Relation steht, ansonsten scheitert es. `nicht_m_verbunden/2` schlägt

fehl, wenn das Prädikat `ist_m_verbunden/2` für irgendein `X` aus `Rest` `true` ist. `nicht_m_verbunden/2` ist erfolgreich, wenn es kein solches `X` aus `Rest` gibt, das `ist_m_verbunden(X,Rest)` erfüllt.

```
sind_alle_m_verbunden([]).
sind_alle_m_verbunden([X|R]) :-
  ist_m_verbunden(X,R),
  sind_alle_m_verbunden(R).
```

`sind_alle_m_verbunden(Z)` ist erfolgreich, wenn alle Elemente aus `Z` untereinander M-verbunden sind. Es prüft mit `ist_m_verbunden/2` für jedes Element `X`, ob dieses zu allen Elementen der Restliste M-verknüpft ist; es scheitert, wenn es ein beliebiges `X` gibt, das zu irgendeinem Element aus der Restliste nicht M-verbunden ist.

Betrachten wir als Testbeispiel folgenden Graphen, bei dem man nicht unmittelbar durch bloßen Augenschein sämtliche Cliquen erkennen kann.

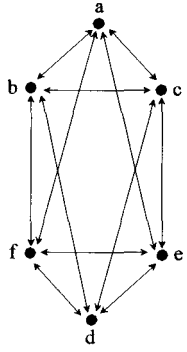


Abb.9.8: Testgraph aus Freeman/Libhart (1986: 121) zur Cliquenermittlung

An diesem Testbeispiel zeigen Freeman/Libhart (1986), daß der Algorithmus von Rattinger unvollständig ist. Die Autoren berichten, daß das Verfahren von Rattinger in eine Endlosschleife mündet, während von Collani feststellt, daß der Rattinger-Algorithmus in seiner Version nur vier der insgesamt acht Cliquen entdeckt. Wir benutzen diesen Graphen nun als "experimentum crucis" für unser Verfahren:

```
?- clique.

Mit Regel  34  abgeleitet:   clique([f,e,d])     -> Faktenbasis
Mit Regel  34  abgeleitet:   clique([f,b,d])     -> Faktenbasis
Mit Regel  34  abgeleitet:   clique([e,a,c])     -> Faktenbasis
Mit Regel  34  abgeleitet:   clique([e,d,c])     -> Faktenbasis
Mit Regel  34  abgeleitet:   clique([e,f,a])     -> Faktenbasis
Mit Regel  34  abgeleitet:   clique([d,c,b])     -> Faktenbasis
Mit Regel  34  abgeleitet:   clique([c,b,a])     -> Faktenbasis
Mit Regel  34  abgeleitet:   clique([b,f,a])     -> Faktenbasis
```

Das Programm findet alle vorhandenen acht Cliquen, allerdings in einer nicht unbeträchtlichen Laufzeit von etwas mehr als zwei Minuten (486/66 MHz unter DOS/Windows). Der hier vorgestellte Algorithmus dürfte trotzdem effizienter sein als

der von Harary/Ross. Er ist um so schneller, je weniger Relationen pro einzelnen Knoten abgehen. Die Effizienz dieses Verfahrens bedürfte einer eigenen Untersuchung, was an dieser Stelle nicht geleistet werden kann, aber hiermit zur Diskussion gestellt wird. Insbesondere könnte man sich auch Heuristiken und Optimierungsverfahren überlegen, die den Algorithmus beschleunigen. Als erstes müßte man z.B. unterbinden, daß Cliquen mehrmals abgeleitet werden, deren Eintrag in die Faktenbasis ja nur vom Regelinterpreter verhindert wird. Für unsere Modellinputs, bei denen die Knotenmenge nicht allzu umfangreich ist, liefert das Verfahren aber ein vernünftiges Zeitverhalten.

Mit der allgemeinen Cliquenermittlungs-Regel ist die T-Graph-Regel von S. 204 eigentlich überflüssig, da alle M-Cliquen (in T-Graphen) auch Cliquen sind. Man beachte aber, daß das Cliquensuchen in T-Graphen aufgrund der Eigenschaften transitiver Graphen wesentlich effizienter und schneller vor sich geht als in Nicht-T-Graphen.

9.4.3.2 Status und Balancegrad

In transitiven Graphen bringt die Ordnung von Personen bzw. M-Cliquen durch A bzw. A* unterschiedliche Popularität zum Ausdruck. Je höher die Ebene, der man angehört, desto mehr Wahlen erhält man. In Cliquen hoher Ebene erhält man neben erwiderten Wahlen aus der eigenen M-Clique vor allem auch unerwiderte aus allen M-Cliquen niedriger Ebenen. Umgekehrt gilt, daß man um so mehr Wahlen abgibt, je niedriger die Ebene ist, der man angehört, insbesondere richtet man Wahlen an alle Personen aus höheren Ebenen (Hummell/Sodeur 1987: 156-157).

Ist die Transitivität verletzt, existieren keine durch A* geordnete M-Cliquen. Es sind in der Regel aber ebenso Cliquen im eben behandelten nicht-transitiven Sinn vorhanden, die eine unterschiedliche Anzahl von R-Wahlen von Akteuren außerhalb der eigenen Clique erhalten. Man kann also auch in Nicht-T-Graphen Cliquen mit unterschiedlicher Popularität und verschiedenem Status unterscheiden, wenn man den Status einer Clique über die Anzahl der erhaltenen Wahlen von Elementen definiert, die nicht in der eigenen Clique enthalten sind. Wir führen deshalb in Hinblick auch auf spätere Belange ein Prädikat `status/3` ein, in dem wir die Anzahl der erhaltenen Wahlen festhalten, welche eine Clique erhält. An dritter Argumentstelle tragen wir noch die Größe der Clique ein, so daß

```
status([a,b,c,d],10,4)
```

bedeutet: Die Clique `[a,b,c,d]` ist vom Status zehn und der Größe vier.

```
regel(35,cliquen,status(Clique,Status,Groesse),
   [Clique, ' mit ', Groesse, ' Mitgliedern wird gewaehlt von ',
   Status, ' Aussenelementen: ', Waehlerliste])
   :-
   leite_ab(clique(Clique)),
   finde_alle_waehler_von(Clique,Clique,Waehlerliste),
   card(Waehlerliste,Status),
   card(Clique,Groesse).

?- leite_ab(status(X,Y,Z)).
```

leitet zunächst eine Clique ab (entweder als Faktum-Eintrag oder über die Cliquen-Regel). Das Prädikat `finde_alle_waehler_von/3` sucht für eine Clique alle Wähler der Clique, die nicht in der Clique enthalten sind.

```
finde_alle_waehler_von([],_,[]).

finde_alle_waehler_von([C|Rest],Cl,Neuwahl) :-
   finde_alle_waehler_von(Rest,Cl,Wahllist),
   finde_waehler_von(C,Cl,W),
   append(W,Wahllist,Neuwahl).

finde_waehler_von(X,Cl,L) :-
   findall(Y,leite_ab(r(Y,X)),L),
   not member(Y,Cl).
```

Das hier nicht aufgelistete Prädikat `card(Waehlerliste,Status)` berechnet die Länge der Wählerliste (also Anzahl der Wähler) und `card(Clique,Groesse)` die Länge der Cliquenliste (also Größe der Clique).

Als letztes inhaltliches Prädikat wollen wir noch den Transitivitäts- oder Balancegrad nach Definition 7 angeben. Der Balance-Index ist maximal, wenn es keine intransitiven Tripel gibt, er ist minimal, wenn es ausschließlich intransitive Tripel gibt. Ist G der Graph, i(G) die Anzahl intransitiver Tripel und n die Zahl der Knoten in G, dann ist der Balance-Index TRX (vgl. Def. 7, S.191):

$$TRX(G) := 1 - \frac{6 \cdot i(G)}{n \cdot (n-1) \cdot (n-2)}$$

Dieser Definition entspricht die folgende PROLOG-Regel.

```
regel(36,graph,balancegrad(Balance),['Anzahl intransitiver Tripel:',
   'Anzahl aller Tripel: ',Alle_Tripel])
   :-
   zaehle(triade(_,intransitiv),In_zahl),
   knoten(X),
   zaehle(X,Kn_zahl),
   Alle_Tripel is Kn_zahl * (Kn_zahl-1) * (Kn_zahl-2)/6,
   Balance is 1 - In_zahl/Alle_Tripel.
```

Man kann nun eine Gruppe G zu zwei verschiedenen Zeitpunkten t und t' mit t' > t vergleichen und den Balancegrad bestimmen. Nach dem Fundamentalgesetz der HL-Theorie müßte der Balancegrad von G zum späteren Zeitpunkt t' größer/gleich dem früheren Zeitpunkt t sein. Das bislang entwickelte Programm kann als Analyseinstrument hierfür eingesetzt werden.

An dem praktischen Beispiel dieser Implementierung können wir nochmals einige abstrakte Diskussionspunkte von Teil I veranschaulichen:

- Definitionen in der Theorie werden direkt in das Computermodell übernommen, ohne daß eine Einführung der von Lindenberg (S.51ff.) angesprochenen Spezialisierungen, Operationalisierungen oder Randbedingungen nötig ist. Die Theorie wird unmittelbar in das Programm auf einer "hohen theoretischen Ebene" abgebildet und es erfolgt auch keine Prozessionalisierung im Sinn von Lindenberg.

Das Schema von Abb.3.2, S.55 kann wie folgt auf dieses Modell angewendet werden: Es gibt einen theoretischen und davon deutlich getrennten nicht-theoretischen Programmteil, wobei die Menge der theoretischen Zusatzannahmen bislang leer ist. Beim theoretischen Programmteil gilt folgendes:

- Manchen Definitionen entsprechen mehrere PROLOG-Regeln (z.B. Regel für T-Graph);
- Manche Definitionen/Theoreme sind nicht repräsentiert (z.B. Reflexivität von R);
- Manchen - dem theoretischen Teil zuzurechnenden - PROLOG-Regeln entsprechen keine Definitionen im Modell (z.B. Clique, Status.)

Klassifikatorisch entspricht das Modell in der Einteilung von Abelson (Abb. 2.9, S.34) der Simulation elementaren sozialen Verhaltens (wenn man die übliche intendierte Anwendung auf soziale Beziehungen verwendet). Bei Gebrauch des Schemas von Troitzsch (S.35) liegt mit dem Programm ein dynamisches, deterministisches, qualitatives Mehrebenenmodell vor.

9.4.4 Erklärungskomponente

Die Erklärungskomponente soll die mit den Regeln hergeleiteten Fakten rechtfertigen und die Arbeitsweise des Modells durchsichtiger machen. Die für die Erklärungen verwendeten Regeln werden direkt dem Programmcode entnommen ("direkte Erklärungen", vgl. Kap. 4.1.2, S.69 und für die vorliegende Implementierung Schnupp/Nguyen Huu 1987, Kap. 5).

Als erstes müssen wir Hilfsprädikate definieren, die eine *leserliche* Ausgabe der definierten Regeln erzeugen. Die Ausgabe der Regeln kann analog wie bei ART mit den folgenden Prädikaten erfolgen. Das Hauptprädikat zur Regelausgabe `gib_regel_aus(Nr,Typ)` wird dabei um eine Argumentstelle (Regel-)`Typ` erweitert.

```
gib_regel_aus(Nr,Typ) :-
   clause(regel(Nr,Typ,Konkl,Beding),Ursach),
   schreibe_liste(['Regelnr.',Nr]),
   write('WENN '),
   schreibe_ursachen(Ursach),
   write(' DANN '),
   schreibe_konklusion(Konkl),
   nl,nl.
```

`schreibe_ursachen/1` und `schreibe_konklusion/1` geben rekursiv Bedingungen und Konklusionen aus. Da wir im Regelkörper nicht immer `leite_ab`-Prädikate verwenden konnten, sondern auch frei verwendete wie `member/2` oder `findall/3`, wird die Ausgabe von `leite_ab`-Prädikaten auf die Ausgabe frei verwendeter Prädikate zurückgeführt.

```
schreibe_konklusion(Konkl) :-
   write(Konkl).

schreibe_ursachen(leite_ab(R)) :-                % leite_ab-Prädikate
   schreibe_ursachen(R).
```

```
schreibe_ursachen(not leite_ab(R)) :-
   schreibe_ursachen(not R).

schreibe_ursachen(leite_ab(R) ',' Ursachen) :-
   schreibe_ursachen(R ',' Ursachen).

schreibe_ursachen(not leite_ab(R) ',' Ursachen) :-
   schreibe_ursachen(not R ',' Ursachen).

schreibe_ursachen(R ',' Ursachen) :-          % frei verwendete Prädikate
   write(R),
   write(' UND '),
   schreibe_ursachen(Ursachen).

schreibe_ursachen(not R ',' Ursachen) :-
   write(' NOT '),
   write(R),
   write(' UND '),
   schreibe_ursachen(Ursachen).

schreibe_ursachen(R) :-
   write(R).

schreibe_ursachen(not R) :-
   write(' NOT '),
   write(R).
```

Das Prädikat `regeln/0` listet alle Regeln auf.

```
regeln :-
   alle_regeln(1).

alle_regeln(Nr) :-
   gib_regel_aus(Nr,_),
   N1 is Nr + 1,
   alle_regeln(N1).
alle_regeln(_) :-
   write('\nAll rules').
```

Bei der Regelausgabe ergibt sich allerdings das Problem, daß PROLOG für die Variablen in den Regeln Nummern verwendet, was in der Arbeitsweise von PROLOG begründet ist.

```
?- regeln.

Regelnr. 1
WENN r(_980,_984) UND r(_984,_980) DANN m(_980,_984)

Regelnr. 2
WENN r(_1296,_1300) UND  NOT r(_1300,_1296) DANN a(_1296,_1300)

Regelnr. 3
WENN    NOT  r(_1628,_1632)  UND    NOT  r(_1632,_1628)  UND    DANN
n(_1628,_1632)
```

```
Regelnr. 4
WENN r(_2024,_2028) UND r(_2028,_2064) UND  NOT r(_2024,_2064) DANN
graph(t_graph,false)
....
```

Dieses Problem bestand bereits bei der Erklärungskomponente von ART. Die Beschreibung der Lösung wurde dort aber übergangen, da in ART der für dieses Modell entwickelte Instantiierungmechanismus übernommen wurde.

Wir müssen nun also dafür sorgen, daß statt der unleserlichen Variablennummern leserliche Variablennamen ausgegeben werden, was nicht ganz trivial ist. Für die Instantiierung der Variablen bei der Regelausgabe verwenden wir folgendes Verfahren: wir führen eine "Variablenliste" ein, deren Elemente aus PROLOG-Sicht Konstanten sind:

```
variablenliste([x,y,z,u,v,w,x1,x2,x3,x4]).
```

Vor jeder Ausgabe eines Regelprädikats erzwingen wir eine Instantiierung aller in der Regel vorkommenden PROLOG-Variablen mit Konstanten aus der Variablenliste. Variable gleichen Namens innerhalb einer Regel müssen dabei mit genau einer Variablen aus der Konstantenliste instantiiert werden. Für eine bestimmte Regel vergebene Konstante werden temporär in der Faktenbasis gespeichert, und bei Auftreten einer neuen, noch nicht instantiierten Variablen verwenden wir die nächste aus der Variablenliste. Nach Ausgabe der Regel löschen wir wieder alle Einträge.

```
instantiiere(X) :-                      % instantiere X, z.B. rel(X,X,Y)
   X=..L,                               % L -> [rel,X,X,Y]
   pruefe_instanz(L,L1),                % L1 ->[rel,x,x,y]
   X=..L1.                              % X -> rel(x,x,y)

pruefe_instanz([],[]).                  % Liste leer -> fertig
pruefe_instanz([X|R],[X|R2]) :-         % Pruefe Kopf X der Liste [X|R]
   var(X),                              % X ist noch nicht instantiiert
   variablenliste(L),                   % L ist die Variablenliste
   member(V,L),                         % V ∈ L
   not schon_vergeben(V),               % V ist noch nicht vergeben
   X = V,                               % Identif. V mit X
   asserta(schon_vergeben(V)),!,        % trage die neue, vergebene Konstante ein
   pruefe_instanz(R,R2).                % Rufe pruefe_instanz mit Restliste auf
pruefe_instanz([X|R],[X|R2]) :-         % Pruefe Kopf X der Liste [X|R]
   nonvar(X),                           % X ist schon instantiiert
   pruefe_instanz(R,R2).                % Rufe pruefe_instanz mit Restliste auf
```

Beispielsweise liefert

```
?- Rel = p(X,Y,Z), instantiiere(Rel).
Rel = p(x,y,z)
```

und

```
?- Rel = p(X,Y,Z,X,X,Y), instantiiere(Rel).
Rel = p(x,y,z,x,x,y).
```

Nun müssen nur noch das `gib_regel_aus`-Prädikat und die Ausgabe-Prädikate leicht modifiziert werden. Am Anfang des Regelkörpers von `gib_regel_aus/2` erfolgt ein Dummy-Eintrag von `schon_vergeben/1` in die Datenbasis (ansonsten gibt es eine Fehlermeldung beim erstmaligen Aufruf von `pruefe_instanz/2`), am Schluß des Regelkörpers werden alle Einträge von `schon_vergeben/1` entfernt (mit dem Built-in-Prädikat `abolish/1`), und schließlich wird das Backtracking verhindert.

```
gib_regel_aus(Nr,Typ) :-
   clause(regel(Nr,Typ,Konkl,Beding),Ursach),
   asserta(schon_vergeben(dummy)),
   schreibe_liste(['Regelnr.',Nr]),
   write('WENN '),
   schreibe_ursachen(Nr,Ursach),
   write(' DANN '),
   schreibe_konklusion(Konkl),
   abolish(schon_vergeben(_)),
   nl,nl,!.
```

Vor jeder Ausgabe eines Prädikats muß nun nur noch das `instantiiere`-Prädikat aufgerufen werden, so daß die obengenannten Prädikate wie folgt zu ändern sind.

```
schreibe_konklusion(Konkl) :-
   instantiiere(Konkl),
   write(Konkl).

schreibe_ursachen(R ',' Ursachen) :-
   instantiiere(R),
   write(R),
   write(' UND '),
   schreibe_ursachen(Ursachen).

schreibe_ursachen(not R ',' Ursachen) :-
   write(' NOT '),
   instantiiere(R),
   write(R),
   write(' UND '),
   schreibe_ursachen(Ursachen).
...
```

Damit erhalten wir folgende, besser lesbare Regelausgabe.

```
?- regeln.

Regelnr. 1
WENN r(x,y) UND r(y,x) DANN m(x,y)

Regelnr. 2
WENN r(x,y) UND  NOT r(y,x) DANN a(x,y)

Regelnr. 3
WENN  NOT r(x,y) UND  NOT r(y,x) DANN n(x,y)

Regelnr. 4
WENN r(x,y) UND r(y,z) UND  NOT r(x,z) DANN graph(t_graph,false)
```

```
Regelnr. 5
WENN  NOT graph(t_graph,false) DANN graph(t_graph,true)

Regelnr. 6
WENN r(x,y) UND r(y,z) UND  NOT r(x,z) DANN
triade([x,y,z],intransitiv)

Regelnr. 7
WENN r(x,y) UND r(y,z) UND r(x,z) DANN triade([x,y,z],transitiv)
...
```

Das Prädikat `regel/1` gestattet die Anzeige von Regeln bestimmter Nummern oder bestimmter Gruppen.

```
regel(X) :-
   numeric(X),
   gib_regel_aus(X,_).
regel(X) :-
   not numeric(X),
   regel_gruppe(Nr,X).

?- regel(cliquen).

Regelnr. 8
WENN graph(t_graph,true) UND
     knoten(x) UND
     member(y,x) UND
     findall(z,leite_ab(m(y,z)),u) UND
     not u = []
DANN m_clique([y|u])

Regelnr. 9
WENN m_clique(x) UND
     m_clique(y) UND
     sind_alle_a_waehler_von(x,y)
DANN a_stern(x,y)

Regelnr. 34
WENN knoten(x) UND
     member(y,x) UND
     zyklus(y,z,m) UND
     sind_alle_m_verbunden(z) UND
     differenz(x,z,u) UND
     nicht_m_verbunden(u,z)
DANN clique(z)

Regelnr. 35
WENN clique(x) UND
     finde_alle_waehler_von(x,x,y) UND
     card(y,z) UND
     card(x,u)
DANN status(x,z,u)
```

Die Cliquenregeln sind für Benutzer, die nicht mit PROLOG vertraut sind, zum Teil kryptisch und könnten noch schöner geschrieben werden. Im Extremfall können die

Original-Regelprädikate bei der Ausgabe ersetzt werden durch umgangssprachliche Phrasen, so daß z.B. Regel 8 wie folgt ausgegeben wird.

```
WENN der Graph transitiv ist UND
     x die Knotenmenge ist UND
     y Element von x ist UND
     u alle M-Waehler von y sind UND
     u nicht leer ist
DANN bildet u mit y eine M-Clique.
```

Allerding bleibt das Manko bestehen, daß in den Cliquenregeln Hilfsprädikate verwendet werden bzw. algorithmisch zu interpretierende Prädikate wie `card/2`, die eine durchgängig deklarative Interpretation erschweren.
Nach diesen Vorarbeiten können, ganz analog zu ART, die `wie`-Prädikate zur Schlußrechtfertigung der abgeleiteten Fakten definiert werden.

```
wie(X) :-
   faktum(X,[Nr|Rest]),
   schreibe_liste(Rest),
   gib_regel_aus(Nr,_),
   write('Also: '),
   write(X).
```

Das 0-stellige Prädikat `wie` ohne Argument rechtfertigt den zuletzt hergeleiteten Fakt. Da die Daten immer mit `asserta/1` an den *Anfang* der Faktenbasis eingereiht werden, genügt eine einmalige Instantiierung des ersten `faktum`-Eintrags.

```
wie :-
   faktum(X,_),
   wie(X).
```

Schließlich soll es möglich sein, die einzelnen Schritte der rückwärtsverkettenden Inferenzen mit einem "Trace" zu verfolgen. Der Trace ist optional und wird über einen Trace-Schalter realisiert, der ein- und ausgeschaltet werden kann (vgl. Savory 1988: 131):

```
tron(nein).
t_ein :-
   retract(tron(_)),
   asserta(tron(ja)).
t_aus :-
   retract(tron(_)),
   asserta(tron(nein)).
```

Voreingestellt ist der ausgeschaltete Trace, der als PROLOG-Fakt in der Datenbasis steht. `t_ein` schaltet den Trace ein, `t_aus` schaltet ihn aus. Die beiden oben angegebenen `leite_ab`-Prädikate des Regelinterpreters müssen hierzu durch folgende vier Clausen ersetzt werden.

```
/* ********* leite_ab für ausgeschalteten Trace ********** */

leite_ab(X) :-
   tron(nein),
   check(X),
   faktum(X,_).

leite_ab(X) :-
   tron(nein),
   regel(Nr,_,X,Ursach),
   not schon_vorhanden(X),
   schreibe_liste(['Mit Regel ',Nr,' abgeleitet:  ',X,'  ->
                   Faktenbasis']),
   asserta(faktum(X,[Nr|Ursach])).

/* ******** leite_ab für eingeschalteten Trace ********* */

leite_ab(X) :-
   tron(ja),
   check(X),
   schreibe_liste(['Pruefe Faktenbasis: ',X]),
   faktum(X,_),
   schreibe_liste([X,' vorhanden']).

leite_ab(X) :-
   tron(ja),
   schreibe_liste(['Nicht vorhanden:    ',X]),
   clause(regel(Nr,_,X,Ursach),U),
   schreibe_liste(['Versuche Regel:     ',Nr]),nl,
   gib_regel_aus(Nr,_),nl,
   regel(Nr,_,X,Ursach),
   not schon_vorhanden(X),
   schreibe_liste(['Regel ',Nr,' erfolgreich.']),
   schreibe_liste(['        Praemissen:']),
   schreibe_liste(Ursach),
   schreibe_liste([X,'    -> Faktenbasis']),
   asserta(faktum(X,[Nr|Ursach])).
```

Die ersten beiden Clausen werden bei Ableitungen mit *ausgeschaltetem* Trace aktiviert, die letzten beiden bei Ableitungen mit *eingeschaltetem* Trace. Der Regelinterpreter bei ausgeschaltetem Trace ist identisch mit den oben angegebenen `leite_ab`-Prädikaten, der Regelinterpreter bei eingeschaltetem Trace wird lediglich um Ausgabeprädikate erweitert, welche es ermöglichen, die durchgeführten Schlußketten zu verfolgen.[73]

Das Programm kann nun leicht um neue Regeln expandiert werden, ohne daß am Regelinterpreter oder an der Erklärungskomponente etwas verändert werden muß. Konsequenterweise kann der Kern des Programms - das ist das Steuerungsystem in Form von Inferenzmechanismus und Erklärungskomponente - nun für die Implementierung *beliebiger* Theorien verwendet werden. Einzige Voraussetzung ist, daß die theoretischen Aussagen in Regelform angegeben werden und sich an unsere Regel-

[73] Interessant wäre noch eine weitere Ergänzung des Erklärungsmoduls um die Möglichkeit zu fragen, warum ein Fakt *nicht* gültig ist. Auf diesen Erklärungszusatz wird hier aber verzichtet.

syntax halten. Ist dies möglich, so steht automatisch der entwickelte Inferenz- und Erklärungsmechanismus zur Verfügung. Dies ist eine wichtige Feststellung, auf die wir später noch einmal zurückkommen werden.

Weitere Prädikate, die nicht wesentlich für das Modell sind, werden nicht besprochen und können dem Anhang entnommen werden. Dies sind insbesondere Prädikate zum Hinzufügen und Entfernen von Daten und Schlußfolgerungen, abkürzende Hilfsprädikate zum Ableiten und das Prädikat `fakten/0`, welches einen Überblick über die aktuell gültige Faktenbasis gibt. Der Benutzer kann sich mit `hilfe/0` alle sinnvoll verwendbaren Prädikate ausgeben lassen.

9.4.5 Beispiele

Die Arbeitsweise des Programms soll an den folgenden Daten demonstriert werden:

```
r(a,b). r(b,a). r(a,c). r(c,a). r(b,c). r(c,b). r(d,e). r(e,d).
r(f,g). r(g,f). r(k,i). r(k,h). r(k,j). r(h,i). r(h,j). r(h,k).
r(i,h). r(i,k). r(i,j). r(j,h). r(j,i). r(j,k). r(a,d). r(a,e).
r(a,f). r(a,g). r(a,h). r(a,k). r(a,i). r(a,j). r(b,d). r(b,e).
r(b,f). r(b,g). r(b,h). r(b,k). r(b,i). r(b,j). r(c,d). r(c,e).
r(c,f). r(c,g). r(c,h). r(c,k). r(c,i). r(c,j). r(d,f). r(d,g).
r(e,f). r(e,g). r(h,f). r(h,g). r(i,f). r(i,g). r(k,f). r(k,g).
r(j,f). r(j,g). r(f,a).
```

Aus der bislang vorgenommenen Implementierung dürfte die grundlegende Deduktionsstrategie als rückwärtsverkettetes Beweissystem klar sein. Das System ist so organisiert, daß sich nach dem Einlesen der Daten jedes über die Regeln definierte Prädikat prüfen und gegebenenfalls unter Benutzung weiterer Regeln ableiten läßt. Wenn wir versuchen, gleich die A*-Relation herzuleiten, so sind hierzu M-Cliquen herzuleiten, was wiederum die Verifizierung des Vorliegens eines T-Graphen und damit die Nicht-Beweisbarkeit von `graph(t_graph,false)` bedingt. Wenn wir den Trace einschalten, so lassen sich die rückwärtsverketteten Inferenzschritte des Systems leicht verfolgen.

```
?- t_ein.
yes

?- a_stern.

Pruefe Faktenbasis:  a_stern(_952,_956)
Nicht vorhanden:     a_stern(_952,_956)
Versuche Regel:      9

Regelnr. 9
WENN m_clique(x) UND m_clique(y) UND sind_alle_a_waehler_von(x,y)
DANN a_stern(x,y)

Pruefe Faktenbasis:  m_clique(_952)
Nicht vorhanden:     m_clique(_952)
Versuche Regel:      8
```

```
Regelnr. 8
WENN   graph(t_graph,true)   UND   knoten(x)   UND   member(y,x)   UND
findall(z,leite_ab(m(y,z)),u) UND not u = [] DANN m_clique([y|u])

Pruefe Faktenbasis:  graph(t_graph,true)
Nicht vorhanden:     graph(t_graph,true)
Versuche Regel:      5

Regelnr. 5
WENN  NOT graph(t_graph,false) DANN graph(t_graph,true)

Pruefe Faktenbasis:  graph(t_graph,false)
Nicht vorhanden:     graph(t_graph,false)
Versuche Regel:      4

Regelnr. 4
WENN r(x,y) UND r(y,z) UND  NOT r(x,z) DANN graph(t_graph,false)

Pruefe Faktenbasis:  r(_5428,_5432)
r(j,g)  vorhanden
Pruefe Faktenbasis:  r(g,_5468)
r(g,f)  vorhanden
Pruefe Faktenbasis:  r(j,f)
r(j,f)  vorhanden
Pruefe Faktenbasis:  r(f,_5468)
r(f,g)  vorhanden
Pruefe Faktenbasis:  r(j,g)
r(j,g)  vorhanden
r(f,a)  vorhanden
Pruefe Faktenbasis:  r(j,a)
Nicht vorhanden:     r(j,a)

Regel  4  erfolgreich.
Praemissen:
r(j,f) r(f,a) not r(j,a)
graph(t_graph,false)      -> Faktenbasis

no
```

Die Herleitung der A*-Relation ist gescheitert, da `graph(t_graph,false)` beweisbar ist und damit die Regeln 5, 8 und 9 nicht feuern. Der Graph ist damit nicht transitiv. Ein intransitives Tripel wurde oben angegeben. Alle intransitiven Tripel erhalten wir mit dem Prädikat `trans_test` (wir schalten den Trace wieder aus, um die Ausgaben nicht zu lang zu gestalten).

```
?- t_aus.

yes

?- trans_test.

Mit Regel 6 abgeleitet:   triade([j,f,a],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([k,f,a],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([i,f,a],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([h,f,a],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([e,f,a],intransitiv) -> Faktenbasis
```

```
Mit Regel 6 abgeleitet:   triade([d,f,a],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([g,f,a],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([f,a,j],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([f,a,i],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([f,a,k],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([f,a,h],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([f,a,e],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([f,a,d],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([f,a,c],intransitiv) -> Faktenbasis
Mit Regel 6 abgeleitet:   triade([f,a,b],intransitiv) -> Faktenbasis

yes

?- balancegrad.

Mit Regel 36 abgeleitet:   balancegrad(0.909091) -> Faktenbasis

yes
```

Offensichtlich ist einzig die Relation r(f,a) verantwortlich für die Intransitivität des Graphen. Wir entfernen nun diese Relation und prüfen den Graphen erneut (zur automatischen Generierung transitiver aus intransitiven Graphen vgl. Kap. 10.1). Das Löschen von einzelnen Fakten ist möglich mit dem hier nicht weiter beschriebenen Prädikat `entferne/1`. Der Aufruf des Prädikats `entferne(X)` hat zur Folge, daß neben dem eigentlich zu löschenden Faktum `X` automatisch *alle bislang hergeleiteten Schlußfolgerungen entfernt werden*. Andernfalls würde die Faktenbasis inkonsistent werden. Der Nachteil ist, daß nun alle interessierenden Deduktionen nochmals abgeleitet werden müssen, was mit wachsender Faktenmenge zeitaufwendiger wird. Eleganter wäre es natürlich, mit nicht-monotonem Schließen zu arbeiten und nur die von den entfernten Fakten *abhängigen* Schlüsse zurückzunehmen (für eine erste Einführung in nicht-monotones Schließen vgl. z.B. Tanimoto 1990: 273-277 und die dort gegebenen Literaturhinweise). Wir wollen diese Komplikation aber hier vermeiden.

```
?- entferne(r(f,a)).

r(f,a)  aus der Datenbasis entfernt
Entferne  triade([f,a,b],intransitiv)
Entferne  triade([f,a,c],intransitiv)
Entferne  triade([f,a,d],intransitiv)
Entferne  triade([f,a,e],intransitiv)
Entferne  triade([f,a,h],intransitiv)
Entferne  triade([f,a,k],intransitiv)
Entferne  triade([f,a,i],intransitiv)
Entferne  triade([f,a,j],intransitiv)
Entferne  triade([g,f,a],intransitiv)
Entferne  triade([d,f,a],intransitiv)
Entferne  triade([e,f,a],intransitiv)
Entferne  triade([h,f,a],intransitiv)
Entferne  triade([i,f,a],intransitiv)
Entferne  triade([k,f,a],intransitiv)
Entferne  triade([j,f,a],intransitiv)
Entferne  graph(t_graph,false)
Bestehende Schlussfolgerungen entfernt. OK.
```

```
yes

?- t_graph.

Mit Regel 5 abgeleitet:   graph(t_graph,true) -> Faktenbasis

yes

?- trans_test.

yes

?- wie.

r(x,y) r(y,z) r(x,z) fuer alle  x,y,z
Regelnr. 5
WENN  NOT graph(t_graph,false) DANN graph(t_graph,true)

Also: graph(t_graph,true)
yes

?- balancegrad.

Mit Regel 36 abgeleitet:   balancegrad(1.0) -> Faktenbasis

yes
```

Der Graph ist nun transitiv und die Herleitung von M-Cliquen und A*-Relationen muß nun gelingen.

```
?- m_clique.

Mit Regel 1 abgeleitet:   m(k,j)  -> Faktenbasis
Mit Regel 1 abgeleitet:   m(k,h)  -> Faktenbasis
Mit Regel 1 abgeleitet:   m(k,i)  -> Faktenbasis
...
Mit Regel  8  abgeleitet:   m_clique([k,j,h,i])     -> Faktenbasis
...
Mit Regel  8  abgeleitet:   m_clique([g,f])         -> Faktenbasis
...
Mit Regel  8  abgeleitet:   m_clique([e,d])         -> Faktenbasis
...
Mit Regel  8  abgeleitet:   m_clique([c,b,a])       -> Faktenbasis

yes


?- a_stern.

Mit Regel 9 abgeleitet:   a_stern([c,b,a],[e,d])       -> Faktenbasis
Mit Regel 9 abgeleitet:   a_stern([c,b,a],[g,f])       -> Faktenbasis
Mit Regel 9 abgeleitet:   a_stern([c,b,a],[k,j,h,i])   -> Faktenbasis
Mit Regel 9 abgeleitet:   a_stern([e,d],[g,f])         -> Faktenbasis
Mit Regel 9 abgeleitet:   a_stern([k,j,h,i],[g,f])     -> Faktenbasis
```

Die deduzierten Fakten entsprechen folgender grafischer Repräsentation.

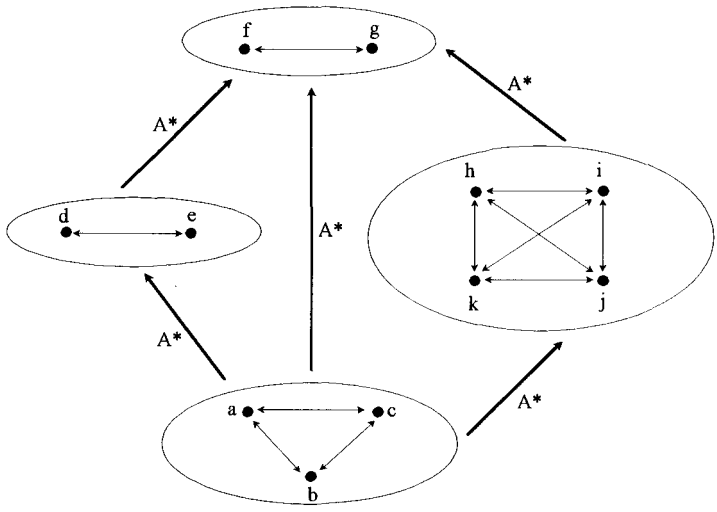


Abb. 9.9: Mit dem Programm deduzierte Beispielstruktur bestehend aus elf Personen und vier M-Cliquen auf drei Ebenen (nach Hummell/Sodeur 1987: 157)

Schließlich können wir noch prüfen, ob ein spezielles Modell bzw. ein spezieller Graph vorliegt.

```
?- graph.
...
Mit Regel  19  abgeleitet:   triadentyp([d,h,f],021u,leer_transitiv)
                              -> Faktenbasis
Mit Regel  33  abgeleitet:   graph(ranked_clusters,true)
                             -> Faktenbasis

yes

?- wie.

graph(t_graph,true), es gibt keine 012-Triaden
Regelnr. 33
WENN graph(t_graph,true) UND  NOT triadentyp(x,012,y) DANN
graph(ranked_clusters,true)

Also: graph(ranked_clusters,true)
```

Ein weiteres Spezial-Modell liegt nicht vor.

Der Graph (b) in Abb. 8.6 (S.171) genügt folgenden Spezialmodellen:

```
?- fakten.

Knoten: [a,b,c,d]

R-Relationen ->  [[a,b],[a,d],[b,a],[b,d],[d,a],[d,b]]

M-Relationen ->  [[d,b],[d,a],[b,d],[b,a],[a,d],[a,b]]
```

```
A-Relationen ->  []

N-Relationen ->  [[d,c],[c,d],[c,b],[c,a],[b,c],[a,c]]

graph(ranked_clusters,true)
graph(structural_balance,true)
graph(clusterable_graph,true)
graph(t_graph,true)
```

Dieser Graph erfüllt also sowohl strukturelle Balance (Cartwright/Harary 1956), den gruppierbaren Graph (Davis 1967), das Ranked-Clustering-Modell (Davis/Leinhardt 1972) und das T-Graph-Modell.

Der Graph von Abb. 8.8 (S.173) wird von Cartwright/Harary (1956) als Beispiel für strukturelle Balance zur Veranschaulichung des Strukturtheorems benutzt. Wenn wir nun versuchen, strukturelle Balance für diesen Graphen auf der Basis des T-Graph-Modells zu berechnen, erhalten wir folgendes Ergebnis.

```
?- graph.

Mit Regel 4 abgeleitet:   graph(t_graph,false) -> Faktenbasis
yes

?- wie.

r(h,g) r(g,f) not r(h,f)
Regelnr. 4
WENN r(x,y) UND r(y,z) UND  NOT r(x,z) DANN graph(t_graph,false)

Also: graph(t_graph,false)
```

Der dem Cartwright-Harary-Modell genügende balancierte Graph wird nicht hergeleitet und ist sogar intransitiv nach dem T-Graph-Modell. Obwohl der Graph nach Cartwright/Harary strukturell balanciert ist, ist er gleichzeitig intransitiv nach dem T-Graph-Modell. Dies scheint ein Widerspruch zu dem Theorem von Holland/Leinhardt zu sein, nach dem das Cartwright/Harary-Modell ein Spezialfall des T-Graph-Modells ist.

Die Ursache für den scheinbaren Widerspruch liegt darin, daß der Graph unvollständig ist, also nicht von jedem Knoten zu jedem anderen Knoten eine definierte Relation verläuft. Wird der Graph derart vervollständigt, daß innerhalb beider Gruppen positive und zwischen den Gruppen negative Relationen verlaufen, wird strukturelle Balance bewiesen. Das Theorem von Holland/Leinhardt gilt offensichtlich nur im Fall von *vollständigen* Graphen. Vollständigkeit wird jedoch im Ursprungsmodell nicht vorausgesetzt. Weder in der Primär- noch in unserer verfügbaren Sekundärliteratur findet sich hierauf ein Hinweis. Zwar ist dieser Sachverhalt formal beweisbar und allein durch optische Inspektion deutlich, unser kleines Programmexperiment hat jedoch auf diese Lücke aufmerksam gemacht.

10. Generierungsmodelle für transitive Strukturen

Die Modelle von Davis, Holland und Leinhardt (D-H-L-Modelle) haben sich als leistungsfähiges Mittel zur Beschreibung struktureller Eigenschaften von Beziehungsnetzen erwiesen. Dennoch haftet diesen Modellen der Mangel an, daß sie nicht erklären können, warum und wie sich solche Strukturen entwickeln: "... ungeachtet der Tatsache, daß die von den Strukturtheoremen hervorgehobenen Merkmale der Vercliquung und Hierarchisierung wichtige Strukturaspekte sind, die unter einer Vielzahl inhaltlicher Gesichtspunkte von Bedeutung sein können, stellt sich die Frage nach den Mechanismen, die die Entstehung, Entfaltung, Stabilisierung und Veränderung solcher Strukturen *erklären* könnten" (Hummell/Sodeur 1987: 160, Hervorhebung von Hummell/Sodeur). Solche Mechanismen finden sich in der Literatur überhaupt nicht oder nur andeutungsweise. Beispielsweise verweisen die obengenannten Autoren vage darauf, daß Vercliquung und Hierarchisierung interaktionstheoretisch, verhaltenstheoretisch oder tauschtheoretisch erklärbar sein könnten, obgleich der Zusammenhang zwischen solchen theoretischen Annahmen und Strukturmodellen wenig stringent ist. Ähnlich fragen sich Cartwright/Harary (1979: 39-40) welche Bedingungen zu Gruppierungstendenzen führen und kritisieren ebenso, daß diese Fragestellung in der empirischen Literatur kaum behandelt wird.

10.1 Hypothesen zur dynamischen Generierung von Balance

Wir wollen nun versuchen, theoretisch interessante Mechanismen, die zur Entstehung von Balance führen, zu implementieren. Dabei sollen Dynamik und Ergebnisse dieser Mechanismen studiert werden, um vielleicht bemerkenswerte theoretische Konsequenzen zu entdecken. Bei der Überführung intransitiver in transitive Strukturen ist insbesondere theoretisch bedeutsam, *welche* (transitive) Strukturen sich unter welchen Hypothesen ergeben. Verwendet werden sollen möglichst Verfahren, die sich aus der *Struktur des Netzes* ergeben und keine oder geringe psychologische oder soziologische Zusatzannahmen enthalten. Dies ist ein wichtiger Punkt, der explizit herausgestellt werden muß. Wir nehmen eine weitgehend *strukturelle Perspektive* ein, in der Individuen *keinen* Einfluß haben und *soziale Kräfte* als determinierend für Verhalten angesehen werden.

Die eben angeschnittenen Fragen werden also zunächst nicht empirisch gelöst, sondern in einem Computational Approach. Falls sich interessante Perspektiven ergeben, können diese dann einer empirischen Prüfung unterzogen werden.

Analog wie bei Abelson/Rosenberg werden nun also Hypothesen eingeführt, *auf welche Weise* eine unbalancierte Struktur in eine balancierte überführt wird. Während in der Abelson-Rosenberg-Theorie das Verfahren, wie diese Überführung erfolgt, Teil des Theoriekerns war, bleibt dies in der Holland-Leinhardt-Theorie ebenso wie bei Heider unspezifiziert. Unsere Hypothesen wären damit also Spezialisierungen des Fundamentalgesetzes von HLT und Zusatzannahmen im Sinn von Kap. 3.1.2, S.51ff.

Abelson/Rosenberg führten als Mechanismus zur Balancegenerierung das Ökonomieprinzip ein, d.h. unbalancierte Strukturen wurden so in balancierte überführt, daß möglichst *wenig* Relationen geändert werden mußten. Analog könnten wir aus einem Nicht-T-Graphen versuchen einen T-Graphen zu erzeugen unter der Bedingung, daß die Kardinalität der Relationenänderungen minimal ist. Dies erscheint aber im Modell von Holland/Leinhardt nicht angebracht. Der Grund ist, daß in ART die Struktur kognitiv in einer Person repräsentiert ist und diese über alle Relationen "verfügt", d.h. die Person hat diese Struktur vollständig unter Kontrolle. Im Modell von Holland und Leinhardt und der intendierten Anwendung auf soziale Gruppen ist die gesamte Struktur aber in der Regel *nicht* in *einem* kognitiven Raum präsent. Auf keinen Fall ist die gesamte Struktur unter *Kontrolle einer* Person, sondern kontrolliert werden nur Relationen, die unmittelbar von einem Akteur ausgehen. Das Prinzip des geringsten Aufwands erscheint hier also völlig unangebracht.

Bevor wir auf theoretische Mechanismen zur Herstellung von Gleichgewicht eingehen, betrachten wir die formalen Möglichkeiten der Balancegenerierung. Ein Graph ist transitiv genau dann, wenn für alle Elemente x,y,z der Datenbasis gilt:
wenn xRy und yRz dann xRz.

Er ist intransitiv genau dann, wenn es ein x,y,z gibt mit xRy, yRz und $\neg$xRz. Zu überprüfen sind alle Tripel des Graphen. Wird ein intransitives Tripel entdeckt, so ist dieses zu korrigieren. Grundsätzlich gibt es zwei Möglichkeiten bei der Korrektur solcher Tripel. Wir können den Wenn-Teil verändern oder den Dann-Teil. Ein Korrektureingriff im Wenn-Teil würde bedeuten, das intransitive Tripel in der Datenbasis so zu ändern, daß die Bedingung der Transitivitätsregel nicht mehr zutrifft. Man erzeugt also eine "leer transitive" Triade, indem entweder xRy oder yRz oder beide aus der Datenbasis *entfernt* werden. Ein Korrektureingriff im Dann-Teil einer intransitiven Triade bedeutet hingegen, die Relation xRz zur Datenbasis *hinzuzufügen*.

Die einfachere Lösung ist zunächst, sich auf das Hinzufügen von Relationen zu beschränken. Abb. 10.1 illustriert, daß die Korrektur intransitiver Triaden durch Hinzunahme von Relationen Auswirkungen auf andere Triaden hat.

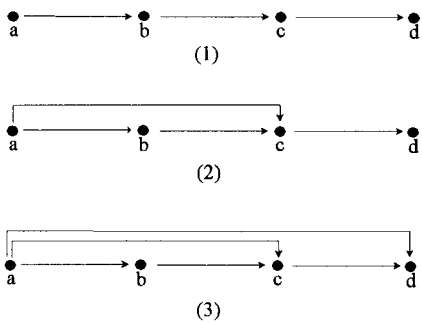


Abb. 10.1: Die Korrektur einer Triade hat "intransitive Auswirkungen" auf transitive Triaden

Die Triade {a,c,d} in Abb. 10.1 (1) ist zunächst leer transitiv (wegen cRd, ¬aRd, ¬aRc), die Triaden {a,b,c} und {b,c,d} sind intransitiv. Wird die Triade {a,b,c} durch Hinzufügen von aRc korrigiert (2), so wird die bislang leer transitive Triade {a,c,d} ebenfalls intransitiv und muß korrigiert werden (3). Die Korrektur intransitiver Tripel kann also dazu führen, daß (leer) transitive Triaden intransitiv werden. Dies hat die Konsequenz, daß die einmalige Prüfung eines Tripels nicht ausreicht.

10.1.1 Einfaches Korrigieren durch Hinzufügen von Relationen

Wenn wir uns beschränken auf Triadenkorrektur durch Hinzufügen von Relationen, so können wir auf der Basis der eben gemachten Überlegungen bereits einen primitiven Korrekturalgorithmus durchführen:
(A1) Suche ein intransitives Tripel xRy, yRz, ¬xRz;
(A2) Füge xRz zur Faktenbasis hinzu;
(A3) Wiederhole Schritt (A1), bis kein intransitives Tripel mehr gefunden wird.

```
balance1 :-
   entferne(alle_schluesse),
   zeige(triade([A,B,C],intransitiv)),
   schreibe_liste_nl([r(A,B),r(B,C),not,r(A,C)]),
   fuege_hinzu(r(A,C)),
   fail.

balance1 :-
   write('Fertig\n').
```

Der Korrekturalgorithmus wird durch den Aufruf des Prädikats `balance1/0` angestoßen. Als erstes müssen wir wieder alle bislang abgeleiteten Schlußfolgerungen zurücknehmen, da die Faktenbasis durch die Hinzunahme von Relationen - analog wie beim Entfernen - ansonsten inkonsistent wird. Das Prädikat

```
entferne(alle_schluesse).
```

nimmt alle abgeleiteten Fakten zurück. Zur Herleitung intransitiver Triaden wird Regel 6 benutzt. Wir verwenden hier nicht das `leite_ab`-Prädikat, da ansonsten jede intransitive Triade in die Faktenbasis eingetragen würde und damit wieder entfernt werden müßte. Die Anordnung der Listenelemente `A, B, C` im Triaden-Prädikat garantiert aufgrund der Struktur des Regelkörpers, daß immer `r(A,B)`, `r(B,C)` und `¬r(A,C)` gilt, so daß jeweils `r(A,C)` mit `fuege_hinzu/1` zur Datenbasis hinzugenommen wird. Durch das `Fail` wird eine Schleife über ein Backtracking zum `zeige/1`-Prädikat realisiert und die nächste intransitive Triade gesucht (die Prädikate `fuege_hinzu/1` und `schreibe_liste_nl/1` sind nur einmal erfüllbar). Der Backtracking-Mechanismus garantiert, daß alle intransitiven Tripel gefunden werden.

Dieses Verfahren ist theoretisch völlig irrelevant, da die Reihenfolge der Triadenkorrektur abhängt von der Reihenfolge des Fakteneintrags in die Datenbasis. Wir wenden uns nun theoretisch interessanteren Korrekturverfahren zu.

10.1.2 Relationen-Addition mit Cliquen als Balance-Initiatoren

Das nun behandelte Verfahren der Triadenkorrektur kann als eine Annahme darüber verstanden werden, warum und wie ein intransitiver Graph in einen transitiven überführt wird. Der Algorithmus realisiert eine Hypothese über die dynamische Entwicklung von Balance. Er prognostiziert genau eine Konfiguration und erhöht damit den von Opp (1984: 33) als mangelhaft kritisierten "empirischen Gehalt" der Theorie.

Kern dieses Korrekturverfahrens ist die Annahme, daß in einem intransitiven Graphen Cliquen die ersten Korrekturträger sind. Dies läßt sich damit begründen, daß in Cliquen der Zusammenhalt besonders groß ist und die Nach-Außen-Wahl eines Mitglieds einen starken Druck auf die anderen Mitglieder ausübt, ebenfalls dieses Außenelement zu wählen. Wenn in Cliquen der Korrekturdruck größer ist als bei Mitgliedern, die keinen Cliquen angehören, so bilden Cliquen die Ausgangsbasis der Veränderung und "infizieren" Nicht-Cliquen-Elemente. Im folgenden benutzen wir den Graphen in Abb. 10.2 als Beispiel.

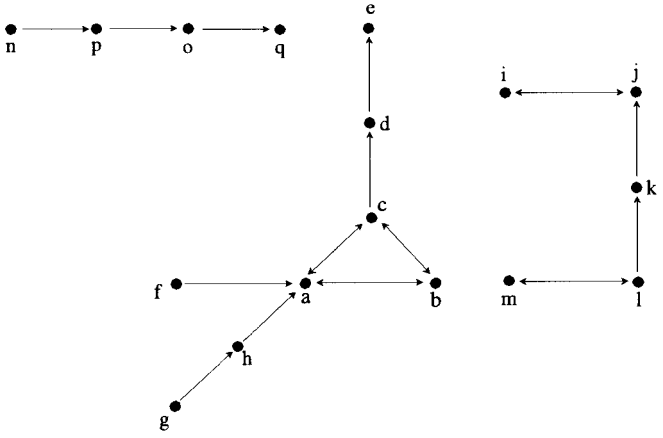


Abb. 10.2: Beispielgraph zur Entwicklung des zweiten Balancegenerierungs-Verfahrens

In Abb. 10.2 wird jedes Mitglied der Dreier-Clique {a,b,c} die Relationen zu d und e korrigieren und dadurch die Knoten f, h und g "infizieren", wodurch diese nun ihrerseits neben den Relationen zu c und b auch die Relationen zu d und e ändern müssen. Durchgeführt wird eine Tiefenkorrektur, d.h. wird eine Relation zu einem Knoten eingeführt, so wird diese Kante "in die Tiefe" weiterverfolgt und geprüft, ob die neu hinzugefügte Kante Intransitivitäten erzeugt usw. In Abb. 10.2 wird bei der Korrektur von Clique {a,b,c} zunächst aRd generiert und der Pfad in die Tiefe weiterverfolgt, d.h. als nächstes wird aRe eingeführt. Erst wenn die Tiefenkorrektur von a abgeschlossen ist, wird der nächste Cliquenknoten b bzw. c untersucht.

Da in einem Graphen normalerweise mehrere Cliquen (in unterschiedlicher Größe) auftreten, brauchen wir eine Regel, bei welcher Clique mit der Korrektur begonnen

werden soll. Wir führen hierzu zwei Parameter ein: die Cliquengröße und den Cliquenstatus.

Wird der Parameter *Cliquengröße* eingeschaltet, so werden die Cliquen in der Reihenfolge ihrer Größe korrigiert. Hierbei hat man die Wahl, mit kleinen Cliquen aufsteigend zu größeren zu beginnen oder umgekehrt, mit größeren absteigend zu kleineren. Wir wählen hier die erste Möglichkeit mit der Begründung, daß mit zunehmender Cliquengröße der Informationsfluß zwischen den einzelnen Mitgliedern schwerfälliger wird und Korrekturen entsprechend zeitverzögert einsetzen. (Die Reihenfolge der Korrekturen innerhalb von Cliquen gleicher Größe überlassen wir der Reihenfolge des Eintrags in der Datenbasis)

Wird der Parameter *Cliquenstatus* eingeschaltet, dann werden die Cliquen in absteigender oder aufsteigender Reihenfolge ihres Status korrigiert. Sinnvoll erscheint hier, mit Cliquen von höherem Status zu beginnen, da in solchen Cliquen ein größerer Druck zur Balanceherstellung existieren dürfte.

Der erste Schritt des Verfahrens - Korrektur von cliquenbedingten Intransitivitäten - ist damit abgeschlossen. Nach der Korrektur der Intransitivitäten, deren Ursache *Mitglieder* von M-Cliquen sind, ändern wir im zweiten Schritt Relationen jener Elemente, die Cliquen wählen, aber selbst *nicht Mitglied* von Cliquen sind. Ausgehend von den Cliquen suchen wir alle Knoten des Graphen, die in R-Relation zu Cliquenmitgliedern stehen, aber nicht selbst Elemente von Cliquen sind. Jeder so entdeckte Knoten wird dabei nach Abarbeitung der direkten Cliquenwähler rückwärts weiterverfolgt. Bei dem Knoten a in Abb. 10.2 sind f und h direkte Wähler von a, so daß zunächst die von f und h ausgehenden Intransitivitäten korrigiert werden. h hat einen weiteren Wähler g, der aber erst bearbeitet wird, wenn alle direkten Cliquenwähler abgearbeitet sind. Ein weiterer direkter Cliquenwähler ist der Cliquenknoten k (wobei k allerdings durch die Tiefensuchstrategie von l, m bereits korrigiert sein kann). Diese Nicht-Cliquenmitglieder werden in der Ordnung entsprechend dem eingestellten Parameter korrigiert. Werden Cliquen in aufsteigender Größe modifiziert, dann werden anschließend jene Nicht-Cliquenmitglieder zuerst korrigiert, die kleinere Cliquen gewählt haben. Das Analoge gilt für den Status.

Im dritten Schritt müssen schließlich noch all jene Elemente bearbeitet werden, die *weder* Cliquenmitglieder sind *noch* - direkt oder indirekt - Cliquenmitglieder wählen. Dies sind die Knoten n, p, o, q in Abb. 10.2.

Zusammengefaßt ergeben sich also folgende Regeln:

- Die Transitivitätskorrektur beginnt in Cliquen mit Tiefensuchverfahren, wobei gilt:
- - Je kleiner die Clique, um so eher beginnt die Transitivitätskorrektur
  - (bzw. alternativ: Je höher der Cliquenstatus, um so eher beginnt die Transitivitätskorrektur).
- Die Transitivitätskorrektur setzt sich fort bei Nicht-Cliquenmitgliedern, die Cliquenmitglieder wählen und Nicht-Cliquenmitglieder, die Nicht-Cliquenmitglieder wählen, welche Cliquenmitglieder wählen usw. Die Änderung findet in der Cliquenreihenfolge statt, die dem eingestellten Parameter entspricht.
- Die Transitivitätskorrektur endet bei Knoten, die weder Elemente von Cliquen sind noch solche Elemente wählen.

Damit erhalten wir folgenden Korrekturalgorithmus:

(A1) Suche alle Cliquen `M`, sortiere sie nach dem Parameter Größe (oder Status) und plätte[74] sie in die Liste `Cl`.

(A2) Instantiiere die Knotenliste `X` und bilde die Differenzmenge `Nicht_Cl = X \ Cl`.

(A3) Instantiiere das Kopfelement `C` von `Cl`.

(A3-1) Korrigiere alle von `C` ausgehenden intransitiven Tripel.

(A3-2) Suche zu `C` alle Knoten `Y`, die Nicht-Cliquenmitglieder sind und `C` wählen, sammle sie in der Liste `L`, und hänge alle Wähler von `Y` an `L` an.

(A3-3) Hänge `L` an die sortierte Cliquenliste `Cl` hinten an.

(A4) Wiederhole Schritte A3 mit der Restliste von `Cl` bis `Cl` leer ist.

(A5) Korrigiere für jedes `k ∈ Nicht_Cl` die von `k` ausgehenden intransitiven Tripel.

Der zweite Korrekturalgorithmus wird mit dem Prädikat `balance2/0` angestoßen.

```
balance2 :-
   parameter_wahl(Typ,Ordnung),
   erzeuge_order_list(Typ,Ordnung,Cl_order),!,
   plaette(Cl_order,Cl),
   entferne(alle_schluesse),
   knoten(X),
   differenz(X,Cl,Nicht_Cl),
   durchsuche_clique(Cl,Cl,[]),
   durchsuche_aussen(Nicht_Cl,Nicht_Cl).
```

`parameter_wahl(Typ,Ordnung)` fragt den Benutzer nach der Reihenfolge, in der die Cliquen sortiert werden sollen. Das Argument `Typ` ist nach der Benutzereingabe instantiiert auf `status` oder `groesse`, das Argument `Ordnung` auf `auf` (für "aufsteigend") oder `ab` (für "absteigend"). Abhängig von diesen Instantiierungen erzeugt `erzeuge_order_list(Typ,Ordnung,Cl_order)` unter Verwendung von `status/3` die sortierte Cliquenliste `Cl_order`. Zum Sortieren benutzen wir das Prädikat `insertsort`, welches "Sortieren durch Einfügen" realisiert und in leicht modifizierter Form Bratko (1987) entstammt.[75]

In unserem Beispiel mit dem Parameter "Größe/aufsteigend" ist `Cl_order` danach instantiiert auf:

```
[[[a,b,c],3],[[m,l],2],[[i,j],2].
```

Das Prädikat `plaette/2` plättet die Liste und entfernt die nicht mehr benötigten Zahlen, welche die Größe der Clique angeben. Die so entstehende Liste `Cl` enthält danach sämtliche Mitglieder von allen Cliquen in sortierter Reihenfolge:

```
[a,b,c,m,l,i,j].
```

---

[74] Unter "Plätten" einer Liste versteht man das Überführen einer verschachtelten Liste - das ist eine Liste, welche als Elemente Listen enthält, die wieder Listen enthalten können usw. - in eine nicht verschachtelte Liste. Das Plätten der Liste `[a,[b,c],d,e,[f,[g,h,i],j,[k,l,[m]]]]` ergibt beispielsweise die Liste `[a,b,c,d,e,f,g,h,i,j,k,l,m]`.

[75] Es gibt eine Vielzahl verschiedener und je nach Problemstellung unterschiedlich effizienter Sortieralgorithmen. Eine Beschreibung und Implementierung der wichtigsten Sortierverfahren findet sich z.B. in Bratko (1987).

Die genaue Arbeitsweise der Prädikate kann dem Anhang entnommen werden.

Als nächstes werden alle bereits abgeleiteten Schlußfolgerungen entfernt. Dies ist einerseits früher nicht möglich, da die hergeleiteten Cliquen und der Status benötigt werden für die sortierte Cliquenliste. Andererseits dürfen die Schlußfolgerungen aber auch nicht beibehalten werden, da Relationen hinzugefügt werden und die Faktenbasis sonst inkonsistent wird.

Nach diesen Vorarbeiten beginnt der eigentliche Algorithmus (A3) - (A5).

Mit der instantiierten Knotenmenge `knoten(X)` bildet das Prädikat `differenz/3` die Differenzmenge `Nicht_Cl` zwischen der Knoten-Grundmenge `X` und der Menge aller Cliquenmitglieder `Cl`, so daß die Grundmenge `X` faktisch partitioniert wird in zwei Teilmengen `Cl` (Cliquenmitglieder) und `Nicht_Cl` (Nicht-Cliquenmitglieder): `X = Cl` $\cup$ `Nicht_Cl` (mit `Cl` $\cap$ `Nicht_Cl` $= \varnothing$).

Im Beispiel von Abb. 10.2 sind die Variablen danach wie folgt unifiziert:

```
X:          [a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q]
Cl:         [a,b,c,i,j,m,l]
Nicht_Cl:   [d,e,f,g,h,k,n,p,o,q]
```

Die nächsten beiden Prädikate übernehmen die Hauptaufgabe, das Absuchen und Korrigieren dieser beiden Teilmengen: `durchsuche_clique/3` untersucht die Cliquen (und deren Anhänger), `durchsuche_aussen/2` die nicht mit Cliquen assoziierte Knotenmenge `Nicht_Cl`.

Das Prädikat `durchsuche_clique/3` arbeitet rekursiv die Menge der Cliquenmitglieder `Cl` ab und besteht aus vier Clausen.

```
durchsuche_clique([],_,_).

durchsuche_clique([C|Cl_Rest],L,L2) :-
   member(C,L2),!,
   durchsuche_clique(Cl_Rest,L,L2).

durchsuche_clique([C|Cl_Rest],L,L2) :-
   test(C,Z),
   korrigiere_wahl_von(C),
   finde_waehler_von(C,L,L1),
   append(Cl_Rest,L1,NeuRest),!,
   durchsuche_clique(NeuRest,L,[C|L2]).

durchsuche_clique([C|Cl_Rest],L,L2) :-
   finde_waehler_von(C,L,L1),
   append(Cl_Rest,L1,NeuRest),!,
   durchsuche_clique(NeuRest,L,[C|L2]).
```

Ist die Liste der Cliquenmitglieder und deren Anhänger leer, sind wir fertig. Die zweite Clause dient zur Verhinderung von Endlosrekursion. Ist ein aktuelles Cliquenelement bereits in der Liste `L2`, in welcher alle bereits geprüften Knoten abgelegt werden, enthalten, so wird dieses Element zunächst nicht untersucht und die Prüfung auf das letzte Vorkommnis in `L2` verlegt.

In der dritten Clause prüft das erste Teilziel `test/2`, ob von dem gerade bearbeiteten Element `C` $\in$ `Cl` Intransitivitäten erzeugt werden. Ist dies nicht der Fall, kommt es

zu einem `Fail`, und es wird die vierte Clause von `durchsuche_clique` aufgerufen. Ist das `test`-Prädikat erfolgreich, behebt `korrigiere_wahl_von/1` alle von dem Element `C` verursachten Intransitivitäten. Das nächste Prädikat `finde_waehler_von/3` generiert alle Wähler von `C`: es wird eine Liste `L1` derjenigen Y gebildet, für die gilt: $Y \in X \wedge Y \notin Cl \wedge r(Y,C)$, also eine Liste derjenigen Elemente, die nicht Cliquenmitglieder sind und `C` wählen. Bei den Cliquenelementen b, c, m, l, i gibt es kein solches Y, da diese nicht von Außenstehenden gewählt werden, d.h. für jedes dieser Elemente ist die Liste `L1` leer. Lediglich a wird von f und h gewählt (`L1` instantiiert auf `[f,h]`) und j von k (`L1` instantiiert auf `[k]`). Die so bei jedem Rekursionsdurchgang entstehende Liste von Wählern der Cliquenmitglieder wird an die Restcliquenliste angehängt. Auf diese Weise wird erreicht, daß zuerst *alle Mitglieder* von Cliquen abgeprüft werden und dann die *Cliquenanhänger*. Man beachte: da `f` und `h` an die Rest-Cliquenliste gehängt werden, werden beim Rekursionsdurchgang mit `h` als Kopfelement die Wähler von h gesucht - dies ist g - und an die Restcliquenliste angehängt. In unserem Beispiel würde sich die Liste der Cliquenmitglieder und deren Anhänger `[C|Cl_Rest]` wie folgt entwickeln:

```
C:  j  Cl_Rest  [i,m,l,b,a,c]
C:  i  Cl_Rest  [m,l,b,a,c,k]
C:  m  Cl_Rest  [l,b,a,c,k]
C:  l  Cl_Rest  [b,a,c,k]
C:  b  Cl_Rest  [a,c,k]
C:  a  Cl_Rest  [c,k]
C:  c  Cl_Rest  [k,h,f]
C:  k  Cl_Rest  [h,f]
C:  h  Cl_Rest  [f]
C:  f  Cl_Rest  [g]
C:  g  Cl_Rest  []
```

Der Algorithmus ist also so konstruiert, daß ausgehend von den Cliquen zuerst alle von Cliquenmitgliedern gewählten Elemente in die Tiefe korrigiert werden und dann alle Wähler von Cliquen in die Breite.

Die Überprüfung der Nicht-Cliquenmitglieder und Nicht-Wähler von Cliquenmitgliedern mit dem Prädikat `durchsuche_aussen/2` funktioniert analog. Betrachten wir noch das Prädikat `korrigiere_wahl_von/1` zur Intransitivitätskorrektur.

```
korrigiere_wahl_von(X) :-
   findall(Kand,test(X,Kand),L),
   entferne_mehrfach(L,Neu),
   add_neu(X,Neu).

add_neu(X,[]).
add_neu(X,[K|R]) :-
   tiefe(X,K),
   add_neu(X,R).
add_neu(X,[_|R]) :-
   add_neu(X,R).
```

```
tiefe(X,El) :-
   test(X,El),
   fuege_hinzu(r(X,El)),
   tiefe(X,Z).

test(X,Z) :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,Z)),
   not (X=Y),
   not (X=Z),
   not (Y=Z),
   not leite_ab(r(X,Z),_).
```

Das Aufrufargument dieses Prädikats ist ein Knotenelement X. Das Prädikat korrigiert alle von X ausgehenden Intransitivitäten. Hierbei wird zuerst in die Tiefe korrigiert, dann in die Breite. Werden zu dem Element X die Knotenelemente [a,b,c] gefunden, zu denen eine R-Relation eingeführt werden müßte (d.h. es gilt xRy, yRa, ¬xRa, das Zwischenelement y interessiert hier nicht), so wird zunächst a korrigiert und alle von a verursachten Intransitivitäten werden weiterverfolgt, bevor zum nächsten Element b weitergegangen wird (siehe oben).
Mit den Daten aus Abb. 10.2 (S.237) ergibt sich folgender Output.

```
?- status.

Mit Regel  1  abgeleitet:   m(l,m)              -> Faktenbasis
Mit Regel  1  abgeleitet:   m(m,l)              -> Faktenbasis
Mit Regel 34  abgeleitet:   clique([m,l])       -> Faktenbasis
Mit Regel 35  abgeleitet:   status([m,l],0,2)   -> Faktenbasis
Mit Regel  1  abgeleitet:   m(i,j)              -> Faktenbasis
Mit Regel  1  abgeleitet:   m(j,i)              -> Faktenbasis
Mit Regel 34  abgeleitet:   clique([j,i])       -> Faktenbasis
Mit Regel 35  abgeleitet:   status([j,i],1,2)   -> Faktenbasis
Mit Regel  1  abgeleitet:   m(c,b)              -> Faktenbasis
Mit Regel  1  abgeleitet:   m(a,c)              -> Faktenbasis
Mit Regel  1  abgeleitet:   m(a,b)              -> Faktenbasis
Mit Regel  1  abgeleitet:   m(c,a)              -> Faktenbasis
Mit Regel  1  abgeleitet:   m(b,a)              -> Faktenbasis
Mit Regel 34  abgeleitet:   clique([b,a,c])     -> Faktenbasis
Mit Regel 35  abgeleitet:   status([b,a,c],2,3) -> Faktenbasis

yes

?- balance2.

Parameter CLIQUEN-STATUS
      (1) Reihenfolge von GERINGEM zu GROSSEN Status
      (2) Reihenfolge von GROSSEM zu GERINGEM Status

Parameter CLIQUEN-GROESSE
      (3) Reihenfolge von KLEINEN zu GROSSEN Cliquen
      (4) Reihenfolge von GROSSEN zu KLEINEN Cliquen

Wahl ==> 3.
```

```
Entferne  m(l,m)
Entferne  m(m,l)
Entferne  clique([m,l])
Entferne  status([m,l],0,2)
Entferne  m(i,j)
Entferne  m(j,i)
Entferne  clique([j,i])
Entferne  status([j,i],1,2)
Entferne  m(b,c)
Entferne  m(c,b)
Entferne  m(a,c)
Entferne  m(a,b)
Entferne  m(c,a)
Entferne  m(b,a)
Entferne  clique([b,a,c])
Entferne  status([b,a,c],2,3)

Bestehende Schlussfolgerungen entfernt. OK.

Erstelement:  j  Rest  [i,m,l,b,a,c]
Erstelement:  i  Rest  [m,l,b,a,c,k]
Erstelement:  m  Rest  [l,b,a,c,k]
Fuege hinzu:  r(m,k)
Fuege hinzu:  r(m,j)
Fuege hinzu:  r(m,i)
Erstelement:  l  Rest  [b,a,c,k]
Fuege hinzu:  r(l,i)
Fuege hinzu:  r(l,j)
Erstelement:  b  Rest  [a,c,k]
Fuege hinzu:  r(b,d)
Fuege hinzu:  r(b,e)
Erstelement:  a  Rest  [c,k]
Fuege hinzu:  r(a,d)
Fuege hinzu:  r(a,e)
Erstelement:  c  Rest  [k,h,f]
Fuege hinzu:  r(c,e)
Erstelement:  k  Rest  [h,f]
Fuege hinzu:  r(k,i)
Erstelement:  h  Rest  [f]
Fuege hinzu:  r(h,c)
Fuege hinzu:  r(h,b)
Fuege hinzu:  r(h,d)
Fuege hinzu:  r(h,e)
Erstelement:  f  Rest  [g]
Fuege hinzu:  r(f,c)
Fuege hinzu:  r(f,b)
Fuege hinzu:  r(f,d)
Fuege hinzu:  r(f,e)
Erstelement:  g  Rest  []
Fuege hinzu:  r(g,a)
Fuege hinzu:  r(g,c)
Fuege hinzu:  r(g,b)
Fuege hinzu:  r(g,d)
Fuege hinzu:  r(g,e)
Erstelement:  q  Rest  [o,p,n,k,h,g,f,e,d]
Erstelement:  o  Rest  [p,n,k,h,g,f,e,d]
Erstelement:  p  Rest  [n,k,h,g,f,e,d]
```

```
Fuege hinzu:  r(p,q)
Erstelement:  n  Rest  [k,h,g,f,e,d]
Fuege hinzu:  r(n,o)
Fuege hinzu:  r(n,q)
Erstelement:  k  Rest  [h,g,f,e,d]
Erstelement:  h  Rest  [g,f,e,d]
Erstelement:  g  Rest  [f,e,d]
Erstelement:  f  Rest  [e,d]
Erstelement:  e  Rest  [d]
Erstelement:  d  Rest  []

yes

?- m_cliquen.

Mit Regel  5  abgeleitet:   graph(t_graph,true)   -> Faktenbasis
Mit Regel  1  abgeleitet:   m(m,l)                -> Faktenbasis
Mit Regel  1  abgeleitet:   m(l,m)                -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([m,l])       -> Faktenbasis
Mit Regel  1  abgeleitet:   m(j,i)                -> Faktenbasis
Mit Regel  1  abgeleitet:   m(i,j)                -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([j,i])       -> Faktenbasis
Mit Regel  1  abgeleitet:   m(c,b)                -> Faktenbasis
Mit Regel  1  abgeleitet:   m(c,a)                -> Faktenbasis
Mit Regel  1  abgeleitet:   m(b,c)                -> Faktenbasis
Mit Regel  1  abgeleitet:   m(b,a)                -> Faktenbasis
Mit Regel  1  abgeleitet:   m(a,c)                -> Faktenbasis
Mit Regel  1  abgeleitet:   m(a,b)                -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([c,b,a])     -> Faktenbasis

yes

?- fakten.

R-Relationen ->
[[o,q],[p,o],[n,p],[m,l],[l,m],[l,k],[k,j],[j,i],[i,j],[d,e],[c,d],
[g,h],[h,a],[f,a],[c,b],[b,c],[c,a],[a,c],[b,a],[a,b],[m,k],[m,j],
[m,i],[l,i],[l,j],[b,d],[b,e],[a,d],[a,e],[c,e],[k,i],[h,c],[h,b],
[h,d],[h,e],[f,c],[f,b],[f,d],[f,e],[g,a],[g,c],[g,b],[g,d],[g,e],
[p,q],[n,o],[n,q]]

M-Relationen ->
[[m,l],[l,m],[j,i],[i,j],[c,b],[c,a],[b,c],[b,a],[a,c],[a,b]]

graph(t_graph,true)
m_clique([m,l])
m_clique([j,i])
m_clique([c,b,a])
```

Das Programm hat unter Beibehaltung der Ursprungscliquen die folgende transitive Struktur generiert.

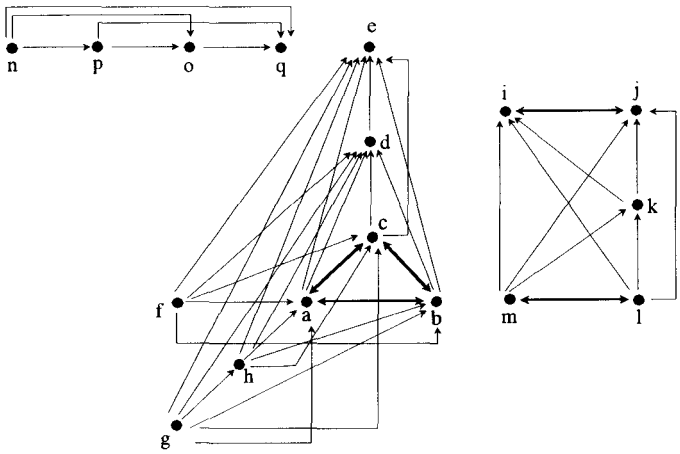


Abb. 10.3: Der nach dem zweiten Balance-Generierungsverfahren korrigierte Graph aus Abb.10.2

Betrachten wir nun den folgenden intransitiven Graphen.

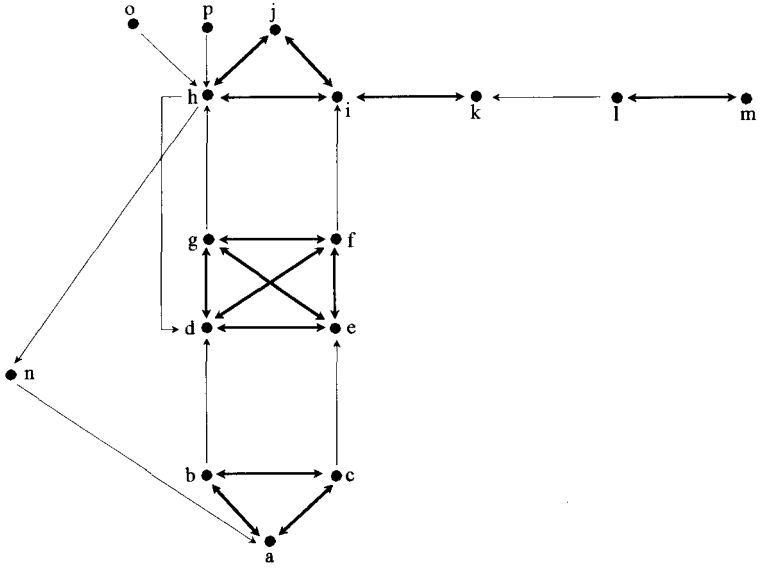


Abb. 10.4: Intransitiver Beispielgraph zum dritten Balancegenerierungs-Verfahren

Hergeleitet werden Cliquen mit folgenden Status- und Größenangaben.

```
status([m,l],0,2)
status([k,i],4,2)
status([j,h,i],5,3)
status([f,g,d,e],3,4)
status([b,a,c],1,3)
```

Nach der Korrektur ergeben sich zwei M-Cliquen:

```
m_clique([m,l])
m_clique([n,a,h,c,b,e,d,i,j,f,g,k]).
```

Die Clique {m,l} bleibt erhalten, während alle anderen Cliquen - und der Nicht-Cliquen-Knoten n - zusammenfallen. Die Vereinigung von Cliquen ist ein unerwünschtes Ergebnis und eine empirische Anomalie. Empirisch fusionieren Cliquen so gut wie nie, allenfalls kommen neue Mitglieder hinzu oder es scheiden einzelne Angehörige aus.

Die Ursache der Cliquenvereinigung läßt sich aus Abb. 10.4 leicht erkennen. Betrachten wir die beiden Cliquen {h,i,j} und {g,f,d,e}. Da h aus der ersten Cliquenmenge d aus der zweiten Menge wählt, müssen mit Transitivität alle Mitglieder der ersten Clique alle aus der zweiten wählen. Da umgekehrt g aus der zweiten Clique h aus der ersten Clique wählt, müssen alle Elemente der zweiten Clique alle Elemente der ersten wählen. Wenn aber alle aus der ersten Clique in R-Relation zu allen Mitgliedern der zweiten Clique stehen und umgekehrt, so bilden beide Ausgangscliquen nach der Transitivitätskorrektur eine einzige Cliquenmenge.

Da die Clique {a,b,c} mit den anderen beiden direkt oder indirekt in analoger Weise verbunden ist, sowie {i,k} direkt mit {i,h,j}, *fallen alle vier Cliquen zusammen.*

Daß der zweite Korrekturalgorithmus die Cliquen in Abb. 10.2 (S.237) erhalten hat, war nur ein zufälliges Ergebnis aufgrund der gegebenen Struktur. Fügt man nur z.B. die beiden Relationen r(b,m) und r(j,f) ein, so werden alle drei Cliquen zusammen mit f und k in eine Clique {m,l,k,i,j,f,a,b,c} vereinigt. Die Ursache für die Cliquenkoinzidenz läßt sich anhand der Graphen in Abb. 10.5 systematischer verdeutlichen.

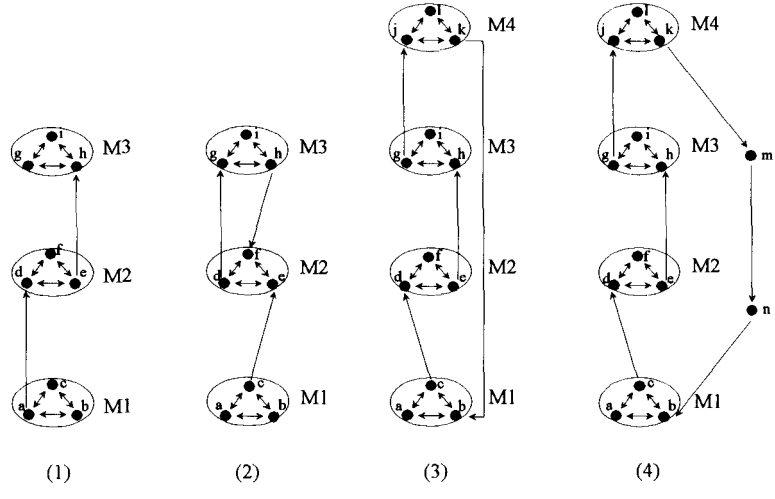


Abb.10.5: Intransitive Graphen mit unterschiedlich verknüpften Cliquen

In Struktur (1) werden die Cliquen bei der Balanceherstellung nicht vereinigt. Die Transitivitätskorrektur durch Hinzufügen von Relationen ergänzt lediglich die Relationen aller M1-Cliquenmitglieder zu M2 und M3, sowie die Relationen von M2-Mitgliedern zu M3-Mitgliedern. Es entsteht eine hierarchische Ordnung zwischen den Cliquen. In Struktur (2) wählt ein Mitglied von M2 eines von M3 und ein Mitglied von M3 eines von M2, so daß beim Korrigieren durch Hinzufügen die Cliquen M2 und M3 zusammenfallen. Die Clique M1 bleibt erhalten. In Struktur (3) sind alle Cliquen durch je eine R-Kante zwischen den Cliquen geordnet und ein Mitglied aus der oberen Clique M4 wählt eines aus der unteren M1. In diesem Fall werden bei dem Korrekturverfahren durch R-Kanten-Hinzunahme alle vier Cliquen vereinigt. Das gleiche gilt auch für Struktur (4), bei der die Wahl des Mitglieds der unteren Clique M1 nicht direkt, sondern über die Zwischenknoten m und n erfolgt.

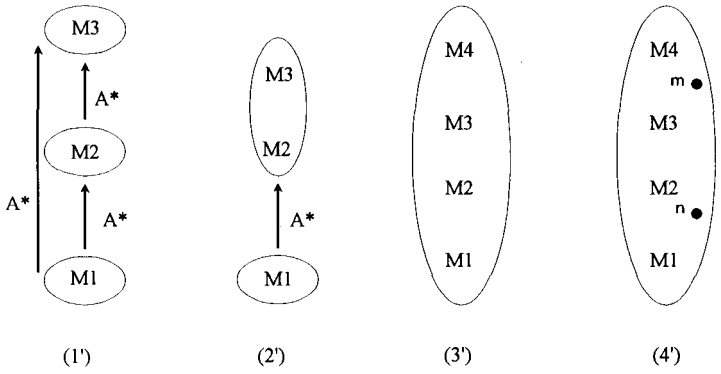


Abb. 10.6: Nach dem zweiten Verfahren korrigierte transitive Graphen aus Abb. 10.5 mit teilweiser Cliquenvereinigung

Die Beispiele suggerieren, daß Cliquen zusammenfallen, wenn es einen *Zyklus* im Sinn von Kap. 8.1.2 (S.168-169) zwischen den Cliquen gibt. In einem Zyklus müssen nach unserer Definition die Knoten und Kanten paarweise verschieden sein. Die Cliquen werden jedoch auch dann vereinigt, wenn die Knoten und Kanten *nicht* paarweise verschieden sind. Beispielsweise fallen in Abb. 10.5 (2) die Cliquen M2 und M3 auch dann zusammen, wenn statt r(h,f) r(g,d) gilt, also das von einem M2-Mitglied gewählte M3-Mitglied die Wahl erwidert. Da auch die Kanten nicht paarweise verschieden sind, liegt auch kein Kreis vor. Wir können jedoch das allgemeinere Konzept des Pfades in einem Graphen benutzen. Zwei Cliquen M_1 und M_2 fallen zusammen, wenn es einen Hinpfad von M_1 zu M_2 und einen Rückpfad von M_2 zu M_1 gibt. Genauer: Wenn M_1 und M_n Cliquen ($M_1 \neq M_n$) sind mit $m_1 \in M_1$ und $m_n \in M_n$ und es gibt einen R-(Hin-)Pfad $P_1 = \langle m_1,...,m_i,...,m_n\rangle$ und einen R-(Rück-)Pfad $P_2 = \langle m_n,...,m_j,...,m_1\rangle$, dann werden die Cliquen M_1, M_n und alle $m_i \in P_1$, $m_j \in P_2$, sowie alle mit m_i, m_j verknüpften Cliquen M_i, M_j vereinigt zu einer einzigen Clique M.

Um im folgenden nicht immer umständlich von Hin- und Rückpfad reden zu müssen, vereinbaren wir eine Abschwächung des Zyklusbegriffs in dem Sinn, daß er

eine Abkürzung für Hin- und Rückpfad ist. Ein R-Hinpfad $P_1 = \langle m_1,...,m_i,...,m_n \rangle$ und ein R-Rückpfad $P_2 = \langle m_n,...,m_j,...,m_1 \rangle$ ist von nun an also ein R-Zyklus $\langle m_1,...,m_i,...,m_n,...,m_j,...,m_1 \rangle$. Alle auf dem Zyklus durchlaufenen Cliquen und Knoten werden vereinigt. Beispielsweise gibt es in Abb. 10.5 (2) einen R-Zyklus ⟨d,g,h,f,d⟩, in (4) einen R-Zyklus ⟨k,m,n,b,c,d,e,h,g,j,k⟩. Würde in (2) statt r(h,f) r(g,d) gelten, lautet ein möglicher R-Zyklus ⟨d,g,d⟩.

Die Cliquen würden nicht zusammenfallen, wenn eine der Zyklen-Relationen fehlen würde, in (3) z.B. die Relation zwischen M4 und M1. Um bei der Korrektur von intransitiven Graphen die Vereinigung von Cliquen zu vermeiden, *müssen also Zyklen zwischen Cliquen unterbrochen werden*. Dies bedeutet, *daß Relationen zurückgenommen werden müssen.*

10.1.3 Erhaltung von Cliquen unter Relationenabbruch

Es soll nun ein Balancegenerierungs-Verfahren entwickelt werden, welches einen intransitiven Graphen in einen transitiven überführt unter der Randbedingung, daß existierende Cliquen möglichst beibehalten werden. Das Zusammenfallen von Cliquen soll durch eine möglichst sparsame Rücknahme von Relationen in Zyklen verhindert werden, die theoretisch begründet wird. Anschließend wird der um diese Relationen reduzierte, aber immer noch intransitive Graph mit dem zweiten Korrekturverfahren in einen balancierten überführt.[76] Theoretisch interessant ist dabei, welche transitive Strukturen sich in welcher dynamischen Entwicklung ergeben.

Bevor wir den eigentlichen Algorithmus entwickeln, sind folgende Fragen zu klären:

- In den Graphen (2-4) von Abb.10.5 verhindert die Annulierung irgendeiner der Zyklen-Relationen die Cliquenvereinigung. Da die Rücknahme jeder Relation in einem Zyklus die Cliquenvereinigung blockiert, muß entschieden werden, welche Relation entfernt wird.
- Da es Zyklen zwischen mehreren Cliquen geben kann, brauchen wir eine Strategie, in welcher Reihenfolge Zyklen gesucht und damit Relationen entfernt werden.

Wir beantworten diese Fragen, indem wir die Relationenrücknahme inhaltlich rechtfertigen. Betrachten wir hierzu die Cliquen {i,j,h} und {d,e,f,g} in dem intransitiven Graphen in Abb. 10.4 (S.245). Der Zyklus zwischen beiden Cliquen kann unterbrochen werden entweder durch Annulierung von r(h,d) oder durch Annulierung beider Relationen r(g,h) und r(f,i). Clique {i,j,h} hat einen höheren Status als Clique {d,e,f,g}. Wir gehen nun von der Annahme aus, daß *die Rücknahme von (Zyklen-) Relationen zu anderen Cliquen immer von der Clique mit höherem Status ausgeht.* Diese Annahme kann gerechtfertigt werden mit Druck oder Sanktionen der anderen Cliquenmitglieder, wenn "niedrigere" Cliquen gewählt werden. In unserem Beispiel hieße das, daß h die Wahl von d stornieren muß, da d einer Clique mit geringerem Status angehört. Die Rücknahme von r(h,d) verhindert aber noch nicht die Vereinigung der Cliquen. Vielmehr gibt es noch einen zweiten Zyklus ⟨h,n,a,b,d,g,h⟩, an

[76] Die umgekehrten Strategie, zuerst Relationen hinzuzufügen und anschließend zu entfernen, erscheint durch die Zusammenlegung von Cliquen wenig sinnvoll. Allenfalls wäre eine Mischstrategie möglich mit abwechselndem Entfernen und Hinzufügen.

dem drei Cliquen beteiligt sind. Gemäß der Regel, nach der in Zyklen Relationen zu untergeordneten Knoten und Cliquen entfernt werden, ist hier r(h,n) zu stornieren. Damit ergibt sich folgende Strategie zum Entfernen von Relationen:

Wir beginnen zunächst mit der Clique M_1 mit höchstem Status und prüfen, ob von M_1 zur Clique mit zweithöchstem Status M_2 Zyklen existieren. Wird ein Zyklus gefunden, so storniert dasjenige $m \in M_1$, das mit seiner (Zyklen-)Relation aus der Clique herausführt, diese Relation. Da von einer Clique mehrere Zyklen ausgehen können, genügt das Testen auf *einen* Zyklus nicht, vielmehr ist solange zu prüfen, bis kein Zyklus mehr existiert. Anschließend werden Zyklen zwischen M_1 und der drittthöchsten Clique M_3 gesucht usw. bis zur untersten Clique M_u.

Wenn die Zyklen-Prüfung für die Clique mit höchstem Status M_1 abgeschlossen ist, wird untersucht, ob von der nächstniederen Clique M_2 Zyklen zu untergeordneten Cliquen $M_3,...,M_u$ ausgehen usw.

Auf diese Weise wird, beginnend mit der Clique mit höchstem Status, jedes Cliquenpaar auf Zyklen abgeprüft und bei positivem Ergebnis die aus der Clique mit höherem Status herausführende R-Kante entfernt.

Der Algorithmus zum Entfernen von Relationen ergibt sich danach wie folgt:

(A1) Suche alle Cliquen M und sortiere sie nach dem Parameter Status/absteigend (alternativ: Größe) in die Liste $S=[M_1,M_2,...,M_n]$;

(A2) Nimm das Kopfelement M_i von S. Ist S=[]. Fertig;

(A2-1) Suche eine Clique M_j mit $M_i \neq M_j$ und $Status(M_i) \geq Status(M_j)$
Fail? Wiederhole (A2) mit Restliste von S;

(A2-2) Finde einen R-Pfad P von $x \in M_i$ nach $y \in M_j$ und von y nach x;
Fail? Wiederhole (A2-1) (d.h. suche eine andere Clique mit kleinerem Status);

(A2-3) Suche Nachbarknoten [..,x,y,..] aus P, so daß $x \in M_i$ und $y \notin M_i$. Entferne xRy aus der Faktenbasis;

(A3) Wiederhole A2 mit Restliste von S.

Anschließend wird der zweite Korrekturalgorithmus aufgerufen.

```
balance3 :-
   parameter_wahl(Typ,Ordnung),
   erzeuge_order_list(Typ,Ordnung,Sortliste),
   bearbeite_cliquen_paar(Sortliste),
   balance2.
```

Das Prädikat `balance3/0` funktioniert analog wie `balance2/0`, mit dem Unterschied, daß als letztes Prädikat `balance2/0` aufgerufen wird.

`erzeuge_order_list(Typ,Ordnung,Sortliste)` erzeugt die nach Statusgröße absteigend sortierte Cliquenliste `Sortliste` (wobei alternativ auch wieder nach Cliquengröße geordnet werden könnte). Das Hauptprädikat

`bearbeite_cliquen_paar(Sortliste)`

führt den obengenannten Algorithmus aus und arbeitet die sortierte Cliquenliste rekursiv ab bis sie leer ist.

```
bearbeite_cliquen_paar([]).

bearbeite_cliquen_paar([[M1,Stat1]|Rest]) :-
   faktum(status(M2,Stat2,_),_),
   ungleich(M1,M2),
   Stat1 >= Stat2,
   suche_zyklus(M1,M2),
   bearbeite_cliquen_paar(Rest).

bearbeite_cliquen_paar([_|Rest]):-
   bearbeite_cliquen_paar(Rest).
```

Zu jeder Clique M1 aus der sortierten Cliquenliste wird eine von M1 verschiedene Clique M2 mit kleinerem Status gesucht. Mit diesen beiden instantiierten Cliquen wird `suche_zyklus/2` aufgerufen, welches einen Zyklus zwischen M1 und M2 sucht. `suche_zyklus(M1,M2)` scheitert, wenn kein Zyklus gefunden wird, d.h. es erfolgt ein Backtracking zum `faktum`-Prädikat, und es wird die nächste Clique M2 gesucht. Gibt es keine solche mehr, scheitert die zweite `bearbeite_cliquen_paar`-Clause, und es wird die dritte `bearbeite_cliquen_paar`-Clause aufgerufen, die sich selbst wieder mit der Rest-Cliquenliste aufruft.
`suche_zyklus/2` sucht einen Zyklus zwischen *Cliquen* M1 und M2:

```
suche_zyklus([X1|M1],[X2|M2]) :-
   einbahn_pfad(X1,X2,Pfad,r),
   schreibe_liste(['Pfad von ',X1,' nach ',X2 ,' gefunden:',Pfad]),
   einbahn_pfad(X2,X1,Pfad2,r),
   schreibe_liste(['Pfad von ',X2,' nach ',X1 ,' gefunden:',Pfad2]),
   finde_cliquen_uebergang(Pfad,[X1|M1],Rel),
   write('Cliquenuebergang: '),write(Rel),nl,
   entferne_relation(Rel),
   schreibe_liste([Rel,' geloescht ',' Clique']),
   suche_zyklus([X1|M1],[X2|M2]).
```

Da *innerhalb* von Cliquen von jedem beliebigen Element ein Zyklus zu allen anderen Elementen existiert, genügt es für die Suche nach Zyklen zu einer anderen Clique, irgendein beliebiges Element einer Clique zu betrachten. Einfachheitshalber benutzen wir die Kopfelemente der Cliquen. Das Prädikat `einbahn_pfad/4` sucht einen - im Unterschied zu `pfad/4` nur einmal erfüllbaren - R-Pfad zwischen dem Kopfelement `X1` der M1-Clique mit gleichem oder höherem Status und dem Kopfelement `X2` der M2-Clique.

```
einbahn_pfad(X,Y,Pfad,r) :-
   pfad(X,Y,Pfad,r),!.
```

Damit ein Zyklus vorliegt, muß ein Pfad zurück von `X2` nach `X1` existieren. Ist ein Zyklus gefunden, so muß die richtige R-Kante entfernt werden. Die zu entfernende R-Kante ist ein Teil des Hinwegs von `X1` zu `X2`, also im `einbahn_pfad`-Prädikat an dritter Argumentstelle in der Liste `Pfad` enthalten. Es muß nun genau diejenige R-Kante gefunden werden, die aus der M1-Clique *herausführt*. Im Graphen von Abb. 10.4 kann `suche_zyklus` z.B. mit irgendeinem Rückweg den Hinweg `[i,j,h,n,a]` in der Pfadliste gesammelt haben. Es gilt nun, aus dieser Liste die-

jenigen Nachbarknoten `[X,Y|Rest]` zu finden, so daß X ∈ M1 und Y ∉ M1. Dies wäre in der obigen Liste das Knotenpaar `[h,n]`.
Das Prädikat `finde_cliquen_uebergang/3` übernimmt die Aufgabe, aus der Pfadliste die zu entfernende Relation zu bestimmen.

```
finde_cliquen_uebergang([X,Y|Rest],Clique,r(X,Y)) :-
   member(X,Clique),
   not member(Y,Clique),!.

finde_cliquen_uebergang([X,Y|Rest],Clique,R) :-
   finde_cliquen_uebergang([Y|Rest],Clique,R).
```

Das Prädikat `finde_cliquen_uebergang([X,Y|Rest],Clique,r(X,Y))` arbeitet die Pfadliste rekursiv ab und ist (einmal) erfolgreich, nämlich genau dann, wenn es zwei Nachbarknoten `X`, `Y` findet, so daß `X` Mitglied von `Clique` ist und `Y` nicht Mitglied von `Clique`.

Mit `entferne_relation(Rel)` als drittletzter Clause von `suche_zyklus/2` wird die gefundene Relation `r(X,Y)` aus der Faktenbasis ausgetragen. Im obigen Beispiel wäre `Rel` mit `r(h,n)` instantiiert und wird aus der Datenbasis gelöscht.

Sind alle Prädikate erfolgreich abgearbeitet, erfolgt ein rekursiver Aufruf von `suche_zyklus/2` mit den *gleichen* Cliquenlisten. Dies klingt nach Endlos-Rekursion, ist aber keine: da im letzten Schritt die Relation aus der Faktenbasis entfernt wurde, kann es nicht mehr den gleichen Hinweg geben. Dies bedeutet, daß es *entweder* einen *anderen Hinweg* gibt - dieser Weg enthielte dann die zweite zu löschende Relation - *oder* es gibt *keinen Hinweg* mehr zu einer anderen Clique. Im letzten Fall scheitert die zweite Clause von `suche_zyklus/2`. Der Aufruf der dritten Clause `bearbeite_cliquen_paar` führt zum rekursiven Aufruf der gleichen Clause mit der Restliste der nach Status geordneten Cliquenliste, und die Prozedur beginnt mit der Clique mit nächst niederem Status erneut.
Mit dem Graphen aus Abb. 10.4 (S.245) ergeben sich folgende Relationen-Annulierungen, -Additionen und M-Cliquen.

```
?- balance3.

Pfad von  k  nach  j  gefunden: [k,i,j]
Cliquenuebergang: i,k
r(i,k)  geloescht

Pfad von  j  nach  f  gefunden: [j,i,h,d,g,e,f]
Pfad von  f  nach  j  gefunden: [f,e,g,h,i,j]
Cliquenuebergang: h,d
r(h,d)  geloescht

Pfad von  j  nach  f  gefunden: [j,i,h,n,a,c,b,d,g,e,f]
Pfad von  f  nach  j  gefunden: [f,e,g,h,i,j]
Cliquenuebergang: h,n
r(h,n)  geloescht

Bestehende Schlussfolgerungen entfernt. OK.
```

```
Erstelement:  k  Rest  [i,m,l,b,a,c,j,h,i,f,g,d,e]
Fuege hinzu:  r(k,j)
Fuege hinzu:  r(k,h)
Erstelement:  i  Rest  [m,l,b,a,c,j,h,i,f,g,d,e]
Erstelement:  m  Rest  [l,b,a,c,j,h,i,f,g,d,e]
Fuege hinzu:  r(m,k)
Fuege hinzu:  r(m,i)
Fuege hinzu:  r(m,j)
Fuege hinzu:  r(m,h)
Erstelement:  l  Rest  [b,a,c,j,h,i,f,g,d,e]
Fuege hinzu:  r(l,i)
Fuege hinzu:  r(l,j)
Fuege hinzu:  r(l,h)
Erstelement:  b  Rest  [a,c,j,h,i,f,g,d,e]
Fuege hinzu:  r(b,f)
Fuege hinzu:  r(b,e)
Fuege hinzu:  r(b,g)
Fuege hinzu:  r(b,i)
Fuege hinzu:  r(b,h)
Fuege hinzu:  r(b,j)
Erstelement:  a  Rest  [c,j,h,i,f,g,d,e]
Fuege hinzu:  r(a,d)
Fuege hinzu:  r(a,e)
Fuege hinzu:  r(a,f)
Fuege hinzu:  r(a,g)
Fuege hinzu:  r(a,i)
Fuege hinzu:  r(a,h)
Fuege hinzu:  r(a,j)
Erstelement:  c  Rest  [j,h,i,f,g,d,e,n]
Fuege hinzu:  r(c,d)
Fuege hinzu:  r(c,f)
Fuege hinzu:  r(c,g)
Fuege hinzu:  r(c,i)
Fuege hinzu:  r(c,h)
Fuege hinzu:  r(c,j)
Erstelement:  j  Rest  [h,i,f,g,d,e,n]
Erstelement:  h  Rest  [i,f,g,d,e,n]
Erstelement:  f  Rest  [g,d,e,n,p,o]
Fuege hinzu:  r(f,j)
Fuege hinzu:  r(f,h)
Erstelement:  g  Rest  [d,e,n,p,o]
Fuege hinzu:  r(g,i)
Fuege hinzu:  r(g,j)
Erstelement:  d  Rest  [e,n,p,o]
Fuege hinzu:  r(d,h)
Fuege hinzu:  r(d,i)
Fuege hinzu:  r(d,j)
Erstelement:  e  Rest  [n,p,o]
Fuege hinzu:  r(e,h)
Fuege hinzu:  r(e,i)
Fuege hinzu:  r(e,j)
Erstelement:  n  Rest  [p,o]
Fuege hinzu:  r(n,c)
Fuege hinzu:  r(n,b)
Fuege hinzu:  r(n,d)
Fuege hinzu:  r(n,e)
Fuege hinzu:  r(n,f)
```

```
Fuege hinzu:  r(n,g)
Fuege hinzu:  r(n,i)
Fuege hinzu:  r(n,h)
Fuege hinzu:  r(n,j)
Erstelement:  p  Rest  [o]
Fuege hinzu:  r(p,i)
Fuege hinzu:  r(p,j)
Erstelement:  o  Rest  []
Fuege hinzu:  r(o,i)
Fuege hinzu:  r(o,j)

yes

?- m_cliquen.

Mit Regel  5  abgeleitet:   graph(t_graph,true)      -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([m,l])          -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([j,i,h])        -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([g,f,e,d])      -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([c,b,a])        -> Faktenbasis

yes

?- a_stern.

Mit Regel  9  abgeleitet:  a_stern([m,l],[j,i,h])      -> Faktenbasis
Mit Regel  9  abgeleitet:  a_stern([g,f,e,d],[j,i,h]) -> Faktenbasis
Mit Regel  9  abgeleitet:  a_stern([c,b,a],[j,i,h])    -> Faktenbasis
Mit Regel  9  abgeleitet:  a_stern([c,b,a],[g,f,e,d]) -> Faktenbasis
```

Generiert wurde somit ein T-Graph mit hierarchischen M-Cliquen folgender Struktur.

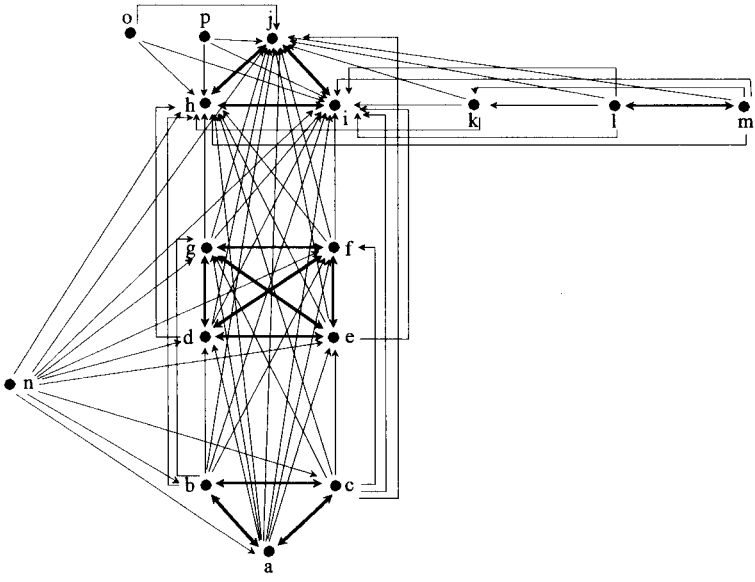


Abb. 10.7: Der mit dem dritten Balancegenerierungs-Verfahren korrigierte Graph aus Abb. 10.4

Alle Ursprungscliquen wurden dabei erhalten mit Ausnahme der Clique {i,k}. Die Bewahrung aller Cliquen mit Ausnahme von {i,k} ist eine theoretische Konsequenz unserer implementierten Annahmen. Wir wollen die Folgerungen nun abschließend systematisch in einem eigenen Kapitel untersuchen.

10.2 Theoretische Konsequenzen

In unserem spezialisierten T-Graph-Modell werden intransitive Strukturen dynamisch in transitive derart überführt, daß zunächst Cliquen und Cliquenumfelder Balance herstellen. Cliquen mit höherem Status nehmen dabei anfangs möglichst sparsam ihre Relationen zu Cliquen mit niederem Status zurück, falls die Gefahr einer Cliquenvereinigung existiert. Anschließend werden Relationen nach der Transitivitätsregel hinzugefügt: zunächst bei Cliquenmitgliedern, dann bei Nicht-Cliquenmitgliedern, die - direkt oder indirekt - Cliquenmitglieder wählen und schließlich bei Personen, die keine Kontakte zu Cliquen haben. Wir wollen nun die dabei entstehende transitive Struktur genauer betrachten und insbesondere die Entwicklung der Cliquen verfolgen.

10.2.1 Die Entwicklung von Cliquen bei Balancetendenz

In der Realität entwickeln sich Cliquen dynamisch.

- Erstens zerbrechen Cliquen manchmal,
- zweitens kommen zu bestehenden Cliquen neue Cliquenmitglieder hinzu und
- drittens entstehen völlig neue Cliquen.

Die Frage ist, ob dies alles unter unseren rein strukturellen Annahmen des *spezialisierten T-Graph-Modells* und der damit verknüpften Balancetendenz *erklärt werden kann*. Ist dies möglich, so lautet die Folgefrage, ob die produzierten Ergebnisse *empirisch sinnvoll* sind. Erscheinen die Folgerungen vernünftig, können drittens die mit dem Computer entdeckten Konsequenzen einer *empirischen Überprüfung* unterzogen werden.

Die erste Frage nach der *Zerstörung von Cliquen* wurde eben angedeutet. Es ist möglich, daß Cliquen zerstört werden und zwar dann, wenn sie unmittelbar über Liaison- oder cocliquale Personen zusammenhängen. Die Zerstörung der Clique {i,k} in Abb. 10.4 (S.245) bzw. Abb. 10.7 (S.253) liegt darin begründet, daß die Cliquen {i,k} und {h,j,i} den Liaisonknoten i haben, wodurch trivialerweise ein Zyklus zwischen beiden Cliquen vorliegt. Da {h,j,i} die Clique mit dem höheren Status ist, wird die zur Clique {i,k} führende Relation r(i,k) annuliert und damit die Clique {i,k} selbst zerstört. Hätte die Ursprungsclique {i,k} einen höheren Status als {h,j,i}, wäre die Clique {h,j,i} eliminiert worden, wobei {h,j} erhalten bliebe.

Analog würde der dritte Balance-Algorithmus in Abb. 9.7 (S.215) in Abhängigkeit vom Cliquenstatus entweder die Dreier-Clique {d,e,f} zerstören oder die Fünfer-Clique {a,b,c,d,e}. Wird die Fünfer-Clique zerstört, bleibt die Dreier-Clique {a,b,c} erhalten. Ebenso würden in Abb. 9.6 (S.210) alle von der Dreier-Clique wegführenden Cliquen eliminiert, wenn die Dreier-Clique den maximalen Status hätte.

Generiert man Balance unter Zyklen-Unterbrechung zwischen Cliquen, bleiben also nicht unmittelbar verbundene Cliquen erhalten. Miteinander verknüpfte Cliquen werden zerstört. Unter der Annahme, daß der Balancedruck bei statushöheren Cliquen größer ist als bei statusniedrigen, werden mit Balancetendenz an statushöhere Cliquen gebundene statusniedrige Cliquen vernichtet.

Ketten von hintereinandergeschalteten Cliquen werden ausgehend vom "Herd" der Clique mit höchstem Status aufgelöst. Diese Clique mit höchstem Status bildet dann den Focus der Wahlen. Die Cliquenkette (1) von Abb. 10.8 wird ausgehend von Clique {c,d} durch die Rücknahme der Cliquenrelationen (2) eliminiert und in die transitive Struktur (3) überführt.

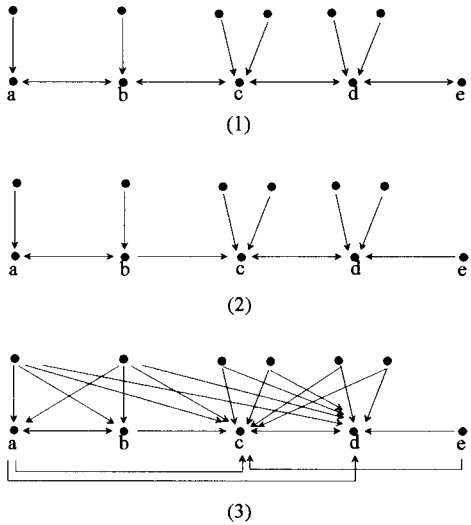


Abb. 10.8: Ketten von hintereinandergeschalteten Zweier-Cliquen

Angenommen, wir führen die Regel ein, daß Cliquen *unter keinen Umständen* zerstört werden dürfen. Diese Regel hätte die Konsequenz, daß unmittelbar benachbarte Cliquen bei der Transitivitätskorrektur zusammenfallen, also aus der obigen Cliquenkette würde eine einzige Fünfer-Clique.

Die Generierung von Balance unter der Randbedingung, Cliquen möglichst beizubehalten, beinhaltet also zwei theoretische Alternativen:

- Entweder direkt über gemeinsame Knoten verbundene Cliquen werden zum Teil aufgelöst oder
- sie fallen in eine einzige Clique zusammen.

Da das Zusammenfallen von Cliquen empirisch die Ausnahme sein dürfte, ist unter Transitivitätstendenz die Auflösung verbundener Cliquen die *einzige* theoretische Alternative. Gegen diese Beobachtung in unserem Modell läßt sich nur das erwähnte empirische Argument anführen, daß der Transitivitätsdruck bei losen Verknüpfungen weniger stark ist.

Die zweite Frage war, *ob unter dem Modell Nicht-Cliquenmitglieder in bestehende Cliquen dynamisch integriert werden können.* Betrachten wir hierzu den folgenden intransitiven Graphen und das Generierungsprotokoll.

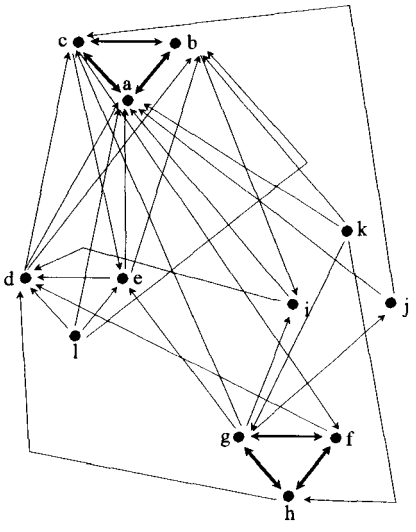


Abb. 10.9: Beispielgraph zur Hinzunahme von Nicht-Cliquenmitgliedern in bestehende Cliquen (nach Moreno aus: Dollase 1973: 107)

```
?- status.

status([g,h,f],3,3)
status([b,a,c],12,3)

yes

?- balance3.

Pfad von  b  nach  g  gefunden: [b,i,d,c,f,g]
Pfad von  g  nach  b  gefunden: [g,j,c,e,b]
Cliquenuebergang: b,i
r(b,i)  geloescht

Pfad von  b  nach  g  gefunden: [b,c,f,g]
Pfad von  g  nach  b  gefunden: [g,j,c,e,b]
Cliquenuebergang: c,f
r(c,f)  geloescht

Bestehende Schlussfolgerungen entfernt. OK.

Erstelement:  b  Rest  [a,c,g,h,f]
Fuege hinzu:  r(b,e)
Fuege hinzu:  r(b,d)
Erstelement:  a  Rest  [c,g,h,f,k,l,e,d]
Fuege hinzu:  r(a,e)
Fuege hinzu:  r(a,d)
Erstelement:  c  Rest  [g,h,f,k,l,e,d,k,j,l,e,d,i]
```

```
Fuege hinzu:  r(c,d)
Erstelement:  g  Rest  [h,f,k,l,e,d,k,j,l,e,d,i,j,d]
Fuege hinzu:  r(g,b)
Fuege hinzu:  r(g,d)
Fuege hinzu:  r(g,a)
Fuege hinzu:  r(g,c)
Erstelement:  h  Rest  [f,k,l,e,d,k,j,l,e,d,i,j,d,k]
Fuege hinzu:  r(h,e)
Fuege hinzu:  r(h,a)
Fuege hinzu:  r(h,b)
Fuege hinzu:  r(h,c)
Fuege hinzu:  r(h,i)
Fuege hinzu:  r(h,j)
Erstelement:  f  Rest  [k,l,e,d,k,j,l,e,d,i,j,d,k,k]
Fuege hinzu:  r(f,e)
Fuege hinzu:  r(f,a)
Fuege hinzu:  r(f,b)
Fuege hinzu:  r(f,c)
Fuege hinzu:  r(f,i)
Fuege hinzu:  r(f,j)
Erstelement:  k  Rest  [l,e,d,k,j,l,e,d,i,j,d,k,k]
Fuege hinzu:  r(k,i)
Fuege hinzu:  r(k,d)
Fuege hinzu:  r(k,f)
Fuege hinzu:  r(k,e)
Fuege hinzu:  r(k,c)
Fuege hinzu:  r(k,j)
Erstelement:  l  Rest  [e,d,k,j,l,e,d,i,j,d,k,k]
Fuege hinzu:  r(l,c)
Erstelement:  e  Rest  [d,k,j,l,e,d,i,j,d,k,k]
Fuege hinzu:  r(e,c)
Erstelement:  d  Rest  [k,j,l,e,d,i,j,d,k,k,l,k]
Fuege hinzu:  r(d,e)
Erstelement:  j  Rest  [l,e,d,i,j,d,k
Fuege hinzu:  r(j,e)
Fuege hinzu:  r(j,b)
Fuege hinzu:  r(j,d)
Erstelement:  i  Rest  [j,d,k,k,l,k,l,e,i,k,k]
Fuege hinzu:  r(i,c)
Fuege hinzu:  r(i,b)
Fuege hinzu:  r(i,e)
Erstelement Aussen:  l  Rest  [k,j,i,e,d]
Erstelement Aussen:  k  Rest  [j,i,e,d]
Erstelement Aussen:  j  Rest  [i,e,d]
Erstelement Aussen:  i  Rest  [e,d]
Erstelement Aussen:  e  Rest  [d]
Erstelement Aussen:  d  Rest  []

yes

?- m_cliquen.

Mit Regel  5  abgeleitet:   graph(t_graph,true)      -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([h,f,g])        -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([e,d,a,b,c])    -> Faktenbasis
```

Die Balancegenerierung mit dem Graphen in Abb. 10.9 liefert als Ergebnis, daß die beiden Cliquen getrennt bleiben, die Ursprungsclique {a,b,c} aber um zwei Elemente *erweitert* wird. Unter unseren Annahmen ist es also möglich, daß Cliquen dynamisch ergänzt werden. Ursachen für solche Erweiterungen sind in unserem Modell immer Zyklen zwischen Cliquenelementen und Nicht-Cliquenelementen, in diesem Fall e und d. Allgemein werden also Nicht-Cliquenmitglieder, die mit Cliquen zyklisch verbunden sind, diesen einverleibt. Existieren von Nicht-Cliquenmitgliedern Zyklen zu mehreren Cliquen, so werden abhängig von dem gewählten Parameter die Nicht-Cliquenmitglieder zuerst entweder den Cliquen höheren Status oder kleinerer Zahl zugeschlagen.

Die Hinzunahme von Personen zu Cliquen, die über Zyklen verbunden sind, scheint bei Face-to-Face-Personennetzen empirisch sinnvoll zu sein. Stellen wir uns eine Zweier-Clique {a,b} vor, wobei a beginnt zu c eine Freundschaft zu knüpfen und c zu d. Unter der Transitivitätsannahme besteht dann die Tendenz, daß auch a zu d sowie b zu c und d Freundschaftskontakte aufnimmt. Angenommen, d wählt nun seinerseits ein Element aus {a,b}. Dann hat dies die Konsequenz, daß über die Zeit hinweg die vier Personen in *eine* Clique zusammenfallen. Wenn wir den Pfeil metaphorisch als "Zuneigungsfluß" interpretieren, so bedeutet das, daß mit dem "Rückfluß" der Zuneigung in die Ausgangsclique die auf dem Pfad liegenden Systemelemente mehr und mehr zusammengeschaltet werden und schließlich in der erweiterten Clique Zuneigung von jedem Element zu jedem anderen Element fließt.

Die dritte Frage war, *ob in dem Modell neue Cliquen quasi "aus dem Nichts" entstehen können.* Betrachten wir hierzu den Graphen in Abb. 10.10

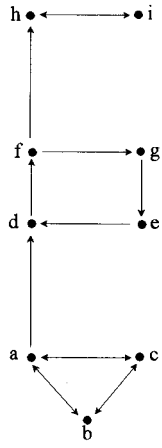


Abb. 10.10: Beispielgraph zur Entstehung neuer Cliquen

```
?- status.

Mit Regel  35  abgeleitet:   status([i,h],1,2)        -> Faktenbasis
Mit Regel  35  abgeleitet:   status([b,a,c],0,3)      -> Faktenbasis
```

Nach der Transitivitätskorrektur ergeben sich folgende Cliquen:

```
?- m_cliquen.

Mit Regel  8  abgeleitet:   m_clique([i,h])          -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([g,e,d,f])      -> Faktenbasis
Mit Regel  8  abgeleitet:   m_clique([c,b,a])        -> Faktenbasis
```

Unter dem Modell entsteht unter Bewahrung der beiden vorhandenen Cliquen eine *neue, vorher nicht existente Clique* {g,e,d,f}. Ursache für die Entstehung neuer Cliquen bei Balancetendenz ist der Zyklus zwischen den vier Knoten {g,e,d,f}. Tendenz zu Balance bedeutet, daß über Zyklen verknüpfte Elemente zu Cliquen ergänzt werden. Dies ist eine allgemein gültige Konsequenz des Modells: *Unter dem T-Graph Modell ist der Zyklus die "Eizelle" für die Entstehung von Cliquen.* Wählt ein Mitglied aus {d,e,f,g} aber eines aus {a,b,c}, wird nach unserem Verfahren die Bildung einer eigenen Clique verhindert, und die Elemente von {d,e,f,g} werden der bestehenden Clique {a,b,c} einverleibt.

Die Entwicklung von Cliquen aus Zyklen scheint bei Personennetzen aus den analogen Gründen wie bei der Cliquenexpansion empirisch einsichtig. Mit dem "Zuneigungsrückfluß" in das Ausgangselement besteht die Tendenz der "Verschweißung" aller "zuneigungsverkoppelten" Elemente in eine Clique. Dieses Zirkelprinzip erinnert an den elektrischen Kurzschluß und die klassischen philosophischen Rückbezüglichkeiten mit dem Urbild der Schlange, die sich in den eigenen Schwanz beißt (vgl. Hughes/Brecht 1978).

Die in unserer künstlichen Welt gemachten Beobachtungen scheinen sinnvoll und müßten nun empirisch getestet werden. In der uns vorliegenden Literatur zur Balancetheorie finden sich zur Cliquenentwicklung keine Hinweise, so daß unsere Ergebnisse als Anregung für empirische Experimente verstanden werden können. Einzige Ausnahme ist die Schlußbemerkung in Holland/Leinhardt (1971: 123): "One substantive interpretation of these results (Auswertung der Soziomatrizen von Davis, Anm.d.Verf.) is that there is a tendency in sociometric data away from imbalance (i.e. transitivity) and that when imbalance does occur it is resolved through transitive closure rather than through the development of vacuous transitivity". Diese Bemerkung ist vereinbar mit unserer Spezialisierung, in der die Transitivitätskorrekturen in erster Linie durch "transitives Schließen" und sehr sparsame Rücknahme der Relationen erfolgt. Sie beantwortet aber nicht die Frage nach den Cliquenüberschneidungen bzw. allgemein zyklen-verknüpften Cliquen. Die zu klärende empirische Frage ist also knapp resümiert: da das T-Graph-Modell zumindest auf den Kern intendierter Anwendungen von Freundschaftsnetzen gut paßt, was passiert mit zyklen-verknüpften Cliquen? Die Alternative unter Balancetendenz kann nur sein: Relationenrücknahme und Trennung oder transitives Schließen und Fusion.

10.2.2 Die Entwicklung starker und schwacher Beziehungen bei Balancetendenz

Granovetter (1973) unterscheidet starke und schwache Beziehungen ("strong ties" und "weak ties") in sozialen Netzen. Die Stärke einer Beziehung ist dabei eine Kombination aus aufgewendeter Zeit, emotionaler Intensität, gegenseitigem Vertrauen sowie dem Ausmaß der wechselseitigen Dienste, die innerhalb der sozialen Verbindung jeweils zum Tragen kommen (Schenk 1984: 72). Zwar sieht Granovetter (1973: 363) soziale Face-to-Face-Gruppen nicht primär als Anwendung für sein Modell, dennoch können Relationen innerhalb von Cliquen als strong ties und Relationen zwischen Cliquen als weak ties im Sinn von Granovetter interpretiert werden. Die durch starke Beziehungen verknüpften Mitglieder innerhalb von Cliquen tauschen nämlich über kurze Distanzen häufiger und intensiver Ressourcen und Informationen aus als Nicht-Cliquenmitglieder bzw. Angehörige verschiedener Cliquen. Andererseits gibt es innerhalb von Cliquen nur wenig Neues, was ausgetauscht werden könnte. Neuigkeiten werden vor allem von jenen Akteuren eingebracht, die nur über schwache Verbindungen verfügen und eine Brückenfunktion zwischen Cliquen einnehmen. In der Graphentheorie bezeichnet man eine Kante als "Brücke", wenn sie den einzigen Pfad zwischen zwei Knoten bildet (Harary/Norman/Cartwright 1965: 198). Beispielsweise nimmt in Abb. 10.11 (1) der Akteur b von Clique {a,b,c,d} eine Brückenfunktion zu e in {e,f,g,h} ein. Wird der Pfeil als Diffusionsrichtung von Information interpretiert, so dringt aus der Clique {a,b,c,d} Information in die zweite Clique ein, nicht aber umgekehrt. Die Schleusung von Information über schwache Beziehungen erreicht dabei eine größere Anzahl von Personen und überwindet weitere soziale Distanzen als über starke Beziehungen.

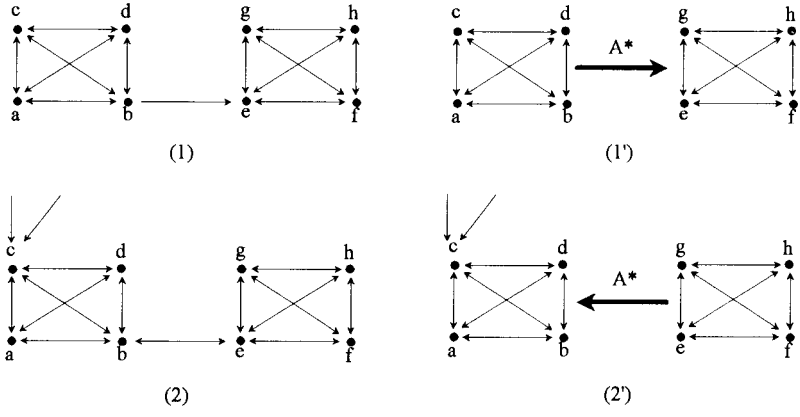


Abb. 10.11 Starke versus schwache Beziehungen und Balancetendenz

Betrachten wir kurz die weniger interessanten starken Beziehungen. Strong ties entstehen im T-Graph-Modell mit der Entwicklung von Cliquen. Werden mit Balancetendenz neue Elemente in bestehende Cliquen integriert, so führt dies zu einer Ausdehnung von strong ties auf die neuen Mitglieder. Zyklen von Nicht-Cliquenmit-

gliedern sind die "Eizellen" für die Entwicklung von Cliquen und damit für die Bildung von strong ties. Starke Beziehungen entstehen unter dem T-Graph-Modell also grundsätzlich aus Zyklen heraus. Interessanter als die Entwicklung starker Beziehungen ist jedoch die der schwachen Beziehungen.

Granovetter betrachtet weak ties als *symmetrisch*, so daß das erste Beispiel von Abb. 10.11 eigentlich keine Anwendung für das Granovetter-Modell ist. Fassen wir vorerst die Verbindung von b zu e in (1) als *asymmetrische* schwache Verbindung auf, so sind die Konsequenzen unter dem T-Graph-Modell folgende: mit der Tendenz zu Transitivität dehnen sich die weak ties auf alle Außenwahlen von Cliquenmitgliedern aus, so daß mit der Einführung der A*-Relation zwischen {a,b,c,d} und {e,f,g,h} jedes Mitglied der ersten Clique zu jedem Mitglied der zweiten Clique eine Relation unterhält, also z.B. Informationen weitergibt (Graph 1' in Abb. 10.11).

Angewendet auf den Graphen in Abb. 10.4 (S.245) würde dies folgendes bedeuten. Die über weak-ties mit allen andern Cliquen verbundene ranghöchste Clique {i,h,j} bekommt mit zunehmender Transitivität immer mehr Informationen und Ressourcen aus allen untergeordneten Cliquen und Nicht-Cliquenmitgliedern. Umgekehrt gibt sie mit ansteigender Transitivität selbst immer weniger Informationen und Ressourcen ab. Im Extremfall der vollständigen Balance empfängt die Clique {i,h,j} in Abb. 10.7 (S.253) nur mehr Informationen/Ressourcen ohne welche zu geben. Die Diffusionswege zu k und den anderen beiden Cliquen wurden mit der Erzeugung von Balance unterbrochen. Bei der rangniedrigsten Clique {a,b,c} ist hingegen gerade das Gegenteil der Fall: sie teilt im Extremfall nur mehr Informationen/Ressourcen aus, bekommt aber keine mehr aus anderen Cliquen. Die Clique {d,e,f,g} in der mittleren Rangposition schließlich erhält von der rangniedrigsten Informationen/Ressourcen, und gibt welche an die ranghöhere ab. Je transitiver ein Graph, um so mehr nehmen also die asymmetrisch interpretierten weak-ties und intercliqualen Informationskanäle von rangniederen zu ranghöheren Cliquen zu und umgekehrt, die von ranghöheren zu rangniederen Cliquen ab. Letztendlich führt dies zu einer hierarchisch verteilten Wissensstruktur: hierarchisch höherstehende Cliquen empfangen mit Transitivitätsneigung immer mehr neue Informationen aus untergeordneten Cliquen, während mit fallender Statushierarchie der Informationsfluß von anderen Cliquen mehr und mehr nachläßt.

Empirisch wurde festgestellt, daß in losen (schwachen) Verbindungen Transitivität weniger zwingend ist als in starken Beziehungen. Diese Beobachtung kann nun mit unserem Modell verknüpft werden, wenn wir weak ties als symmetrisch interpretieren.

Wenn wir, wie Granovetter, die weak ties - also die Verbindungen *zwischen* Cliquen - als *symmetrisch* auffassen (Abb. 10.11 (2)), ergeben sich unter unserem spezialisierten T-Graph-Modell folgende Konsequenzen. In Abb. 10.11 (2) sind die beiden großen Cliquen verbunden durch eine M-Relation, so daß die Knoten b und e ebenfalls als "Clique" betrachtet werden müssen. Information fließt also nun über die Cliquenmitglieder b und e von der ersten zur zweiten Clique und umgekehrt. Wenn wir annehmen, daß die Clique {a,b,c,d} einen höheren Status hat, so annuliert der Algorithmus die Relation von b zu e, während alle Mitglieder von {g,h,e,f} zu allen Elementen von {a,b,c,d} nur mehr einseitige Relationen unterhalten (2' in Abb.

10.11). Dies bedeutet, daß weak ties - symmetrisch interpretiert - mit Balancetendenz aufgelöst werden und in einseitige Relationen transformiert werden. Die Folgerung kann mit den oben gemachten empirischen Feststellungen verknüpft werden, daß lose schwache Verbindungen weniger transitiv sind als starke Verbindungen. Die empirische Feststellung und unsere künstliche Beobachtung haben die gleiche Konsequenz, nämlich, daß symmetrische weak ties *nicht durch transitives Schließen ergänzt werden*: empirisch bleiben sie oft intransitiv, unter dem *Zwang* zu Transitivität werden sie abgebrochen. Dies bedeutet, daß zwischen dem T-Graph-Modell und dem Modell von Granovetter eine stringente Verbindung hergestellt werden kann.

Zusammenfassend läßt sich diese Beobachtung wie folgt interpretieren: Mit der Tendenz zu balancierten Strukturen neigen Cliquen mit hohem Status dazu, symmetrische weak ties aufzulösen, damit Informationsabgabe an andere Cliquen einzuschränken und im Extremfall völlig zu unterbinden. Umgekehrt tendieren in diesem Fall Cliquen mit geringem sozialen Status in Strukturen mit steigender Transitivität dazu, Cliquen mit hohem Status immer mehr mit Information zu versorgen. Allgemein: Mit Balance-Tendenz nehmen in beidseitig verbundenen Cliquen symmetrische weak-ties von Cliquen mit hohem Status zu solchen mit niedrigem ab und umgekehrt nehmen Relationen - die man als asymmetrische weak-ties bezeichnen kann - von Cliquen mit geringem Status zu solchen mit höherem Status zu.

11. Extensionen des T-Graph-Modells

Das ursprüngliche T-Graph-Modell von Holland/Leinhardt und das darauf beruhende regelbasierte Computermodell können weiter ausgebaut und expandiert werden. Wir stellen zunächst theoretische Extensionsmöglichkeiten für das T-Graph-Modell vor und zeigen anschließend an einem konkreten Beispiel, wie das entwickelte Programm für andere Bereiche eingesetzt werden kann.

11.1 Theoretische Erweiterungen des T-Graph-Modells

Zwei theoretische Modifikationen bzw. Erweiterungen des T-Graph-Modells sollen vorgestellt werden, eine in der Literatur existente und eine nicht existente. Eine Abänderung des T-Graph-Modells ist das Modell hierarchischer M~-Cliquen (Johnsen 1985).[77] Dieses Modell ist insofern interessant, als es den sehr strengen Cliquenbegriff, der dem T-Graph-Modell zugrunde liegt, in realistischer Weise abschwächt. Allerdings geht die schöne Eigenschaft der Transitivität verloren. Das Modell geht aus der HL-Theorie dadurch hervor, daß man die intransitive Triade 210 mit zwei M- und einer A-Relation zuläßt. Empirisch wurde beobachtet, daß die intransitive 210-Triade häufiger auftrat als das Zufallsmodell erwarten ließ (Hallinan 1975).

Da die M-Relation im Fall des Vorliegens einer solchen Triade nicht mehr transitiv ist, erhält man auf der Basis von M keine Partition in M-Cliquen mehr. Man kann aber das Konzept der M-Clique wie folgt zum Konzept der M~-Clique abschwächen: Eine M~-Clique ist eine maximale Teilmenge von Knoten, die alle paarweise M~-verbunden sind. Ein Knotenpaar ist M~-verbunden genau dann, wenn es entweder direkt M-verbunden ist oder über eine oder mehrere Zwischenknoten indirekt M-verbunden ist. Die M~-Relation zwischen Knoten ist damit symmetrisch und transitiv und liefert bei vorausgesetzter Reflexivität wieder eine Einteilung in disjunkte Teilgruppen M~. Unter diesen Voraussetzungen gilt das folgende Theorem.

(Struktur-)Theorem (Johnsen 1985: 215)
Für einen Graph ⟨X,R⟩, in dem es nur die Triadentypen 003, 012, 021U, 021D, 102, 210, 030T, 120U, 120D und 300 gibt, gilt:
(1) Die Knoten bilden ein partielle Ordnung hinsichtlich A;
(2) Die Knoten zerfallen hinsichtlich M~- in Teilmengen (M~-Cliquen), so daß gilt:
(2-1) Innerhalb jeder M~-Clique gibt es nur M- und A-Kanten und die Knoten sind hinsichtlich A partiell geordnet;
(2-2) Die Menge aller M~-Cliquen bildet hinsichtlich A* eine partielle Ordnung.

---

[77] Vgl. zum folgenden Hummell/Sodeur (1987: 149-151).

M-Verbindungen verlaufen also weiterhin innerhalb und N-Verbindungen zwischen Cliquen. Im Unterschied zu HLT können nun aber A-Relationen *sowohl* innerhalb *als auch* außerhalb von Cliquen verlaufen, so daß auch *innerhalb von Cliquen eine Hierarchisierung* stattfindet. Abb. 11.1 zeigt eine solche M~-Clique, bei der jedes Mitglied mit jedem anderen direkt oder indirekt M-verbunden ist.

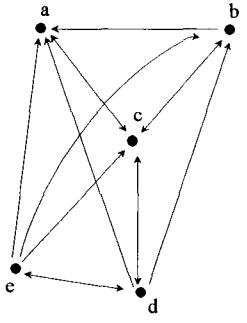


Abb.11.1: Hierarchische M~-Clique

Das Programm kann leicht um das Konzept der M~-Relation und M~-Clique erweitert werden. Die M~-Relation kann z.B. über das Pfad-Prädikat mit folgender Regel definiert werden.

```
regel(37,man, m_schlange(X1,X2),['Es gibt einen M-Pfad ',P,
   ' von ',X1,' nach ',X2])
   :-
   knoten(X),
   member(X1,X),
   member(X2,X),
   not X1 = X2,
   pfad(X1,X2,P,m).
```

Angewendet auf den Graphen von Abb. 11.1 wird folgender Output produziert.

```
?- m_schlange.

Mit Regel  1  abgeleitet:   m(e,d)      -> Faktenbasis
Mit Regel  1  abgeleitet:   m(d,e)      -> Faktenbasis
Mit Regel  1  abgeleitet:   m(c,d)      -> Faktenbasis
Mit Regel  1  abgeleitet:   m(d,c)      -> Faktenbasis
Mit Regel  1  abgeleitet:   m(b,c)      -> Faktenbasis
Mit Regel  1  abgeleitet:   m(c,b)      -> Faktenbasis
Mit Regel  1  abgeleitet:   m(a,c)      -> Faktenbasis
Mit Regel  1  abgeleitet:   m(c,a)      -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(e,d)      -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(e,c)      -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(e,b)      -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(e,a)      -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(d,e)      -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(d,c)      -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(d,b)      -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(d,a)      -> Faktenbasis
```

```
Mit Regel  37  abgeleitet:   m_schlange(c,e)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(c,d)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(c,b)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(c,a)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(b,e)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(b,d)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(b,c)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(b,a)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(a,e)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(a,d)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(a,c)     -> Faktenbasis
Mit Regel  37  abgeleitet:   m_schlange(a,b)     -> Faktenbasis
```

Im allgemeinen Fall, d.h. wenn die Bedingung des Strukturtheorems nicht erfüllt ist, kann die M~-Clique genauso wie die allgemeine Clique bestimmt werden. Da die M~-Clique ebenso wie die Clique über den maximal vollständigen Teilgraph definiert ist und nur die zugrundeliegende Relation eine andere ist, können wir sogar die gleichen Regeln benutzen wie für die Clique. Wir müßten die Cliquen-Prädikate nur allgemeiner umschreiben und ein Argument vorsehen, in dem die zu behandelnde Relation - `m` oder `m_schlange` - eingesetzt wird.

Im speziellen Fall, wenn die Bedingung des Strukturtheorems erfüllt ist, könnte man sich einen effizienteren Algorithmus überlegen, was an dieser Stelle aber nicht mehr weiterverfolgt werden kann.

Wir möchten abschließend noch kurz auf eine gänzlich andere theoretische Erweiterungsmöglichkeit für das Modell vom transitiven Graphen hinweisen. Die von der HL-Theorie (und auch dem Johnsen-Modell) behandelten Daten haben eine sehr einfache Struktur. Die Theorie betrachtet nur empirische Anwendungen, die aus einer Menge von Elementen X bestehen und einer einzigen Grundrelation R. Man kann hier kritisch einwenden, dies sei eine ungemein starke "Reduktion von Komplexität".

Eine naheliegende Ausdehnung wäre nun, nicht nur eine, sondern *mehrere verschiedene Relationen* zuzulassen, also Graphen vom Typ $\tau > 1$. Dies wäre eine weniger starke Vereinfachung, denn wir sind im Alltagsleben in *viele* unterschiedliche Relationen eingebunden. Man könnte zunächst mit zwei Relationen R_1 und R_2 beginnen. Eine mögliche Interpretation für R_1 wäre z.B. wie bisher die Liking-Relation und R_2 könnte in einem bestimmten Kontext als "wissenschaftliche Anerkennung" gedeutet wären (wir abstrahieren hier von der Frage, ob formale Relationen Anwendungen der Transitivitätsregel sind). Untersuchenswert wäre dann, welche Konsequenzen die Anwendung des inhaltlichen Axioms der Transitivitätstendenz auf *beide* Relationen hat. Im Fall von R_2 würde das Axiom bedeuten: wenn x y wissenschaftlich anerkennt und y z, dann besteht für x die Tendenz, auch z wissenschaftlich anzuerkennen. Die Frage ist, welche Strukturen sich mit zwei Relationen entwickeln können. Bilden Personen z.B. eine M_1-Clique, bilden sie nicht zwangsläufig eine M_2-Clique. Schließlich könnte auf der Basis dieses bi-relationalen Ansatzes ein multi-relationales Modell für n Relationen aufgebaut werden. Inwieweit multi-relationale Modelle sinnvoll mit dem single-relationalen T-Graph-Modell vereinbar sind, muß offen bleiben. Gerade hier würde sich jedenfalls der Vorteil des symbolischen Ansatzes zeigen, da mehr als eine Relation in Matrizen nicht mehr so einfach dargestellt werden kann.

11.2 Balancetheorien und Soziometrie

Die primär intendierte Anwendung für D-H-L-Modelle sind Strukturen, wie sie vor allem in der Soziometrie sowie Kleingruppen- und Kommunikationsforschung betrachtet werden. Die Soziometrie ist eine Methode zur Erforschung bestimmter Aspekte der Struktur sozialer Beziehungen in Gruppen (Mayntz/Holm/Hübner 1974: 122, vgl. auch Dollase 1973), so daß sich hier eine direkte Verbindung und Gemeinsamkeit über die Anwendungen ergibt. Soziometrie und Balancetheorien (in der späteren Fassung) betrachten im Prinzip die gleichen empirischen Strukturen. Der Unterschied zwischen ihnen ist, daß Soziometrie eine *Analysemethode* für soziale Beziehungen ist, also ein empirisches Werkzeug, während Balancetheorien *Modelle* sind, die postulieren, daß sich die sozialen Beziehungen in gewisser Weise verändern.

Salopp gesagt, ist das zentrale Anliegen beider Ansätze die Identifikation und Untersuchung bestimmter sozialer Sub-Strukturen. In den D-H-L-Modellen gilt das Interesse hauptsächlich Cliquen als Sub-Strukturen und deren vertikaler und horizontaler Organisation. In der Soziometrie und angrenzenden Kleingruppenforschung interessiert man sich daneben für weitere Sub-Strukturen. Abb. 11.2 gibt einige Teilstrukturen wieder, die z.B. in der Kleingruppenforschung in Hinblick auf Kommunikationseffizienz und Zufriedenheit der Mitglieder untersucht wurden.

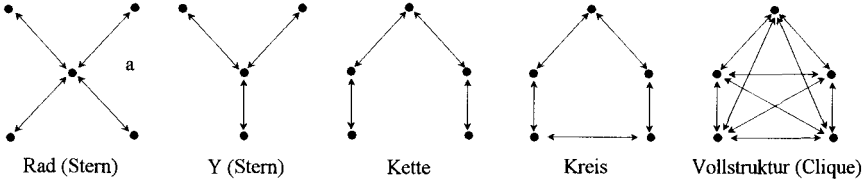


Abb.11.2: Signifikante Sub-Strukturen in der Kommunikationsforschung (nach Dollase 1973: 216)

Merkmal dieser Konfigurationen ist, daß die Zentralität von links nach rechts abnimmt. Das Rad ist die zentralisierteste Formation, in der die Kommunikation über eine einzige Person erfolgt. In weniger zentralisierten Netzen kontrolliert dagegen keine einzelne Person den Informationsfluß, und im Extremfall der Clique kann jeder jeden anderen unmittelbar erreichen. Man beachte, daß unter dem T-Graph-Modell alle Konfigurationen eine Tendenz zur Vollstruktur haben bzw. in unserer Spezialisierung (drittes Verfahren) aufgelöst werden. Allerdings finden sich solche Sub-Strukturen mit Ausnahme der Clique eher in Problemlöse- und Arbeitsgruppen mit extern vorgeschriebenen Relationen, denen man nicht freiwillig angehört (z.B. Betrieb). Gruppen mit extern vorgeschriebenen Relationen sind aber keine Anwendung der HL-Theorie.

In der Soziometrie interessieren neben solch typischen Teilstrukturen bestimmte Positionen von Gruppenmitgliedern, wie der Isolierte (wird von keinem gewählt), der Star (wird von vielen gewählt, die sich untereinander nicht wählen, vgl. Rad und Stern), der Führer oder die oben erwähnten Personen, die zwei Cliquen gleichzeitig angehören ("Liaisons").

Wir haben oben gesagt, daß der Steuerungsteil des entwickelten Programms zur HL-Theorie prinzipiell für beliebige Theorien weiter verwendbar ist. Wir können das Programm aber nicht nur für Theorien verwenden. Vielmehr läßt es sich in diesem Kontext zu einem soziometrischen Analyseinstrument von Netzwerken ausbauen, indem wir einfach entsprechende Regeln hinzufügen. Wir können diese Regeln wie erwähnt ohne Änderungen am Inferenzmechanismus oder an der Schlußfolgerungskomponente hinzufügen und können beide Teile weiter wie oben benutzen. Wir wollen einige Beispiele für soziometrische Definitionen in PROLOG angeben.
Eine Person X ist *isoliert*, wenn sie von keiner anderen Person gewählt wird, es also keinen R-Eintrag in der Faktenbasis mit X an zweiter Argumentstelle gibt.

```
regel(100, position, isoliert(X), ['es gibt kein y, so daß gilt:
   r(y,',X,')'])
   :-
   not leite_ab(r(_,X)).
```

Ein *Star* ist eine Person X, die von sehr vielen (hier: mindestens zwei) gewählt wird, die sich untereinander nicht wählen.

```
regel(101, position, star(X1),[X1, ' wird gewaehlt von ',L,' und
   nicht r(x,y) fuer alle x,y aus ',L])
   :-
   knoten(X),                          % X ist die Knotenmenge und
   member(X1,X),                       % X1 ∈ X und
   findall(Y,leite_ab(r(Y,X1)),L),     % L ist d.Menge aller R-Waehler v. X1 und
   L=[_,_|_],                          % card(L) ≥ 2 und
   not_r_connectet(L).                 % alle Elem. v. L sind nicht R-verbunden
```

Eine *Liaisonperson* ist ein Knoten X, der zwei Cliquen angehört.

```
regel(103, position, liaison(X),[X, ' ist Mitglied von Clique ',M1,
   ' und ',M2])
   :-
   leite_ab(clique(M1)),               % M1 ist eine Clique und
   leite_ab(clique(M2)),               % M2 ist eine Clique und
   not M1 = M2,                        % M1 ≠ M2 und
   member(X,M1),                       % X ∈ M1 und
   member(X,M2).                       % X ∈ M2
```

In ähnlicher Form können die Substrukturen definiert werden, z.B. der *Kreis* als Zyklus (einer Mindestlänge, hier: vier), dessen Elemente nur Nachbarknoten und sonst keine Zyklenelemente wählen.

```
regel(104, substruktur,kreis(Z),['Es gibt einen Zyklus ',Zyklus,
   ' so dass gilt: nicht r(x,y) für alle Nicht-Nachbarkonten'])
   :-
   knoten(X),                          % X ist die Knotenmenge und
   member(X1,X),                       % X1 ∈ X und
   zyklus(X1,Z,m),                     % Z ist ein M-Zyklus von X1 nach X1 und
   Z = [_,_,_,_|_],                    % card(Z) ≥ 4 und
   kreis_test(Z).                      % kreis_test für Z ist wahr
```

`kreis_test(Z)` muß dabei verifizieren, daß innerhalb des Zyklus außer zu den Nachbarknoten keine anderen R-Kanten existieren. Weitere Regeln für soziometrische Definitionen in PROLOG finden sich in Manhart (1987).[78]

[78] Neben Positionen und Substrukturen ist ein weiteres Hauptinteresse in der Soziometrie die Entwicklung aussagekräftiger Indizes, die eine Gruppe und deren Mitglieder charakterisieren. Solche Indizes sollen z.B. Auskunft geben über den inneren Zusammenhalt einer Gruppe, die Integration oder den sozialen Status von Personen. Vgl. hierzu z.B. Dollase (1973).

12. Zum wissenschaftstheoretischen Status von Balancetheorien

In diesem inhaltlich abschließenden Kapitel soll der wissenschaftstheoretische Status der Balancetheorien auf der Basis der strukturalistischen Theorienkonzeption untersucht werden. Damit sind vor allem zwei Aspekte gemeint. Zum einen sollen die Beziehungen *zwischen* den einzelnen Theorien - die *intertheoretischen Relationen* - betrachtet werden und zumindest für ein Theorienpaar formal abgeleitet werden. Zum anderen soll die diachrone oder historisch-systematische Entwicklung der Balancetheorien - die *Theorienevolution* - dargestellt werden. Wie wir sehen werden, zeigt sich bei diesen Fragestellungen eigentlich erst die große Leistungsfähigkeit des strukturalistischen Apparats.

12.1 Intertheoretische Relationen

Wissenschaftliche Theorien treten selten isoliert auf, sondern sind meist eingebunden in ein komplexes Netz theoretischer Strukturen und unterschiedlicher Anwendungen. Bei der Behandlung der Balancetheorien wurde deutlich, daß historisch jüngere Varianten meist als "Verallgemeinerungen" älterer Versionen betrachtet wurden. Der Aufsatz von Cartwright/Harary bezieht sich z.B. schon im Titel darauf, daß strukturelle Balance eine Verallgemeinerung der Heider-Theorie ist. Ebenso betrachten Holland/Leinhardt (1971: 49) ihre Theorie als Generalisierung der bisherigen Ansätze, wobei unklar bleibt, was dies genau bedeutet. Insbesondere wird der logische Zusammenhang zur Heider-Theorie nicht klar herausgearbeitet. Opp kritisiert, daß die Heider-Theorie nur einige heuristische Ideen geliefert hat und kein Versuch gemacht wurde, die genauen logischen Beziehungen zwischen der Heider- und HL-Theorie zu untersuchen: "These relations seemed uninteresting to the authors, for reasons which they do not state" (Opp 1984: 44). Wir wollen nun untersuchen, worin genau diese "Generalisierungsbeziehungen" bestehen, wie sie zu rechtfertigen und formal zu behandeln sind. Wir werden sehen, daß die intertheoretische Relation der Generalisierung in der strukturalistischen Wissenschaftstheorie nicht unmittelbar vorkommt und einem differenzierteren Instrumentarium weicht.

12.1.1 Spezialisierung und Reduktion

Das Studium intertheoretischer Relationen betrachtet Stegmüller (1986: 288-289) als besonders zukunftsträchtiges und erfolgversprechendes Unternehmen. Nicht nur besteht hier ein enormer wissenschaftstheoretischer Nachholbedarf, die bisherige Wissenschaftstheorie stieß auf diesem Gebiet auch drastisch an die Grenzen ihrer Methoden. Der Grund ist, daß in der herkömmlichen Wissenschaftstheorie Theorienvergleich auf einer linguistisch-mikrologischen Ebene stattfand, bei dem Term-für-Term-Vergleiche angestellt wurden. Für viele intertheoretische Relationen ist dies

aber äußerst kompliziert und mühsam. Beim makrologisch-modelltheoretischen Vorgehen des Strukturalisten stehen statt sprachlicher Konstrukte hingegen globale Strukturen im Vordergrund, bei denen von vornherein mit solchen Entitäten wie M, M_p und M_{pp} operiert wird. Bei Vergleichen werden sofort derartige globale Einheiten miteinander in Beziehung gesetzt. Die Erfahrung lehrt, daß ein solcher Umgang mit globalen Strukturen viel einfacher ist als es die entsprechenden mikrologischen Verfahrensweisen sind (Stegmüller 1986: 73-74).

Die Betrachtung mehrerer Theorien, die zueinander in bestimmten Beziehungen stehen, hat einen Wandel in der Sprechweise zur Folge. Was bislang "Theorie" genannt wurde, wird nun als "Theorie-Element" bezeichnet. Dadurch soll klarer ausgedrückt werden, daß das, was im allgemeinenen als wissenschaftliche Theorie gilt, in der Regel aus mehreren, miteinander verknüpften Elementen besteht. Eine intertheoretische Relation ist dann eine Relation zwischen zwei Theorie-Elementen T und T*. Wir schließen uns dieser Sprechweise an und behandeln die vorgestellten Gleichgewichtstheorien als Theorie-Elemente, benutzen aber weiter den Ausdruck "Theorie" auch im Sinn von "Theorie-Element".

In der neueren Literatur (Balzer/Moulines/Sneed 1987) werden fünf grundlegende intertheoretische Relationen genannt: Spezialisierung, Theoretisierung, Reduktion, Äquivalenz und Approximation. Von diesen fünf intertheoretischen Relationen sind in unserem Zusammenhang nur die Spezialisierung und Reduktion relevant.

Spezialisierungs- und Reduktionsrelation können als strukturalistische Explikation und Differenzierung des eingangs erwähnten substanzwissenschaftlichen Generalisierungsbegriffs verstanden werden. Wir betrachten zuerst den einfacheren Fall der Spezialisierung. Unter der *Spezialisierung* eines Theorie-Elementes versteht man das Hinzufügen neuer spezieller Axiome, deren Gültigkeit nur für einen Teilbereich der intendierten Anwendungen behauptet wird. Spezialgesetze werden in der Regel als zusätzliche definitorische Bedingungen zum vorgegebenen mengentheoretischen Prädikat dazugenommen und verschärfen damit das Prädikat. Das neue Theorie-Element ist dann "spezialisierter" als das ursprüngliche Element.

Modelltheoretisch wird die Operation der Spezialisierung auf zwei einfache Voraussetzungen zurückgeführt: Identität zwischen der Klasse der potentiellen und partiellen potentiellen Modelle einerseits und Teilmengenrelation für Modelle und intendierte Anwendungen andererseits. Man schneidet bildlich gesprochen aus der Menge der Modelle jene Teilmenge aus, die durch ein mengentheoretisches Prädikat mit restriktiveren Bedingungen bestimmt wird und auf einen begrenzteren empirischen Bereich angewendet werden soll. Die Spezialisierung läßt sich damit wie folgt definieren (Balzer 1982: 294).

(D-1) Es seien $T = \langle M, M_p, M_{pp}, I\rangle$ und $T^* = \langle M^*, M^*_p, M^*_{pp}, I^*\rangle$ Theorie-Elemente. T* ist eine Spezialisierung von T (abgekürzt: $T^* \sigma T$) gdw
(1) $M^*_p = M_p$
(2) $M^*_{pp} = M_{pp}$
(3) $M^* \subseteq M$
(4) $I^* \subseteq I$

Bei der Spezialisierungsrelation σ bleibt das Begriffsinventar also gleich. Entscheidend ist, daß die speziellere Theorie T* eventuell weniger Modelle hat als die allgemeinere Theorie T. Die Modelle von T* sollen zudem nur für einen Teilbereich I* der ursprünglichen Anwendungen I gelten. Bildlich gesprochen werden aus den Mengen M und I Teilmengen M* und I* durch schärfere Prädikate "herausgemeißelt".

Führt man ausgehend von einem Basiselement T Spezialisierungen T*, T**, T***... durch, die ebenfalls wieder spezialisiert werden können usf., so erhält man ein *Theoriennetz* von Theorie-Elementen in Form einer hierarchischen Ordnung. Ein Theoriennetz, das bestimmten zusätzlichen Bedingungen gehorcht, kann als Baum dargestellt werden (Balzer/Moulines/Sneed 1987: 172-204). In einem solchen Baum ist das oberste (Basis-)Element das allgemeinste Theorie-Element, von dem spezialisiertere Elemente abzweigen. Vielen naturwissenschaftlichen Theorien unterliegen solche Baumstrukturen.

Die zweite und bekannteste intertheoretische Relation ist die *Reduktion*. Grob gesprochen versteht man unter Reduktion das Zurückführen einer einfacheren Theorie T auf eine reichere und komplexere Theorie T*. T* enthält dabei in irgendeiner Form T, so daß T* erfolgreich alle jene Phänomene behandelt, die auch von T gemeistert wurden, sowie zusätzliche, von T nicht bearbeitete Fälle. Wenn wir sagen, daß T reduziert wird auf T*, so ist im folgenden immer angenommen, daß das zweite Relationsargument T* die komplexere oder "bessere" Theorie ist und das erste Argument T die einfachere. Dabei heißt T* die reduzierende, T die reduzierte Theorie.

Reduktionsbeziehungen zwischen Theorien wurden sowohl in der Wissenschaftstheorie als auch in den Substanzwissenschaften ausführlich und kontrovers behandelt. Von einigen Wissenschaftsphilosophen wurde z.B. die Auffassung vertreten, daß Theorien aufgrund intertheoretischer Bedeutungsvarianz einzelner Terme grundsätzlich "inkommensurabel" und damit nicht reduzierbar sind. Bekanntestes Beispiel ist die unterschiedliche Bedeutung des Massebegriffs in der klassischen und relativistischen Mechanik. Eine andere Thematik betraf die vom älteren und neueren Positivismus ausgelöste Debatte, inwieweit ganze Disziplinen auf andere reduzierbar seien, also z.B. Soziologie auf Psychologie und diese wiederum auf Physiologie. Diese Fragestellungen können hier nicht weiter verfolgt werden. Wir halten zumindest die zweite für spekulativ, die erste wird in Balzer/Moulines/Sneed (1987: 313-320) behandelt.

Bei der Reduktionsrelation lassen sich verschiedene Unterfälle unterscheiden, die aber alle in dem gleichen formalen Rahmen behandelt werden können (Balzer/Moulines/Sneed 1987: 253-255). *Historische Reduktion* liegt vor, wenn eine ältere Theorie T auf eine neuere, erfolgreichere Theorie T* reduziert wird. Die wesentlichen Errungenschaften der Vorgängertheorie T müssen dabei nach T* überführt werden, so daß diese auch als Erfolge von T* betrachtet werden können. Bei der *praktischen Reduktion* benutzt man eine einfachere reduzierte Form T einer komplexen Theorie T* aus dem Grund, Schwierigkeiten bei der Anwendung der komplexen Theorie T* auf eine gegebene Problemstellung zu meiden und diese nur grob mit T zu lösen. Ein weiterer Unterschied betrifft exakte und approximative Reduktion. Unter *exakter Reduktion* versteht man vereinfacht gesagt, daß nach entsprechender

Übersetzung die Axiome der komplexeren Theorie T* jene der einfachen Theorie T implizieren. *Approximative Reduktion* liegt hingegegen dann vor, wenn bei diesem Prozeß irgendwelche Näherungen durchgeführt werden müssen. Wir beschränken uns im folgenden auf exakte historische Reduktion.

Die zentrale Forderung beim Reduktionskonzept ist, daß die Gesetze der einfacheren Theorie T aus den Gesetzen der komplexeren Theorie T* logisch ableitbar sein müssen. Fallstudien von naturwissenschaftlichen Theorien, die in der Literatur als reduzierbar angesehen werden, zeigen jedoch, daß eine unmittelbare Ableitung i.a. aus zwei Gründen nicht möglich ist. Erstens verwenden T und T* in der Regel einen *unterschiedlichen Begriffsapparat*. Dies bedeutet, daß eine Verbindung oder ein Brückenprinzip hergestellt werden muß zwischen jenen Termen, die in beiden Theorien nicht identisch vorkommen. Traditionell könnte man auch von der Übersetzung der beiden Theoriesprachen sprechen, aber im Strukturalismus vermeidet man - wie eingangs angedeutet - den Bezug auf Sprachen zugunsten einer modelltheoretischen Sicht. Ein zweiter Grund, warum eine unmittelbare Ableitung meist nicht möglich ist, liegt darin, daß die *Basisgesetze von T* allein zu schwach* sind für die Deduktion der Gesetze von T. In vielen Fällen muß daher T* noch spezialisiert werden. Erst wenn beide Operationen durchgeführt wurden - die Verbindung des Begriffsapparates und die Spezialisierung der komplexeren Theorie - ist in der Regel eine Herleitung möglich.

Formal wird das Reduktionskonzept über eine Übersetzungs- oder Reduktionsrelation ρ definiert, welche die potentiellen Modelle beider Theorien aufeinander bezieht. ρ muß zwei Bedingungen genügen: einmal soll es zu jedem Modell der reduzierten Theorie eine Übersetzung in ein Modell der reduzierenden Theorie geben, und zum anderen sollen sich die Modelleigenschaften der reduzierten Theorie bei Übersetzung ableiten lassen. Wir geben im folgenden eine vereinfachte Definition nach Balzer/Moulines/Sneed (1987: 277).

(D-2) Es seien $T = \langle M, M_p, M_{pp}, I\rangle$ und $T^* = \langle M^*, M^*_p, M^*_{pp}, I^*\rangle$ Theorie-Elemente. ρ reduziert T auf T* (abgekürzt: $T\ \rho\ T^*$) gdw
(1) $\rho \subseteq M^*_p \times M_p$
(2) Für alle x, x*: wenn $x\ \rho\ x^*$ und $x^* \in M^*$, dann $x \in M$

Die zweite Bedingung der Definition drückt dabei modelltheoretisch die zentrale Forderung aus, daß bei Vorliegen der Reduktionsrelation die Gesetze von T aus den Gesetzen von T* ableitbar sind. Im Fall der historischen Reduktion ist eine zusätzliche pragmatische Forderung, daß die neue Theorie T* fähig sein sollte, erfolgreich mit all jenen Anwendungen umzugehen, welche die alte Theorie T schon meisterte. Vereinfacht gesagt wird verlangt, daß jede intendierte Anwendung der reduzierten Theorie T durch die Übersetzungsrelation ρ in eine intendierte Anwendung der reduzierenden Theorie T* übersetzt werden kann (Balzer 1982: 300).

12.1.2 Intertheoretische Relationen zwischen Balancetheorien

Auf der Basis der eben vorgestellten Konzepte können wir nun versuchen, die Frage zu beantworten, welche Relationen zwischen den Gleichgewichtstheorien vorliegen. Da eine genaue systematische Untersuchung aller Theorie-Elemente den Rahmen dieser Arbeit weit sprengen würde und diese auch nicht alle präzisiert wurden, beschränken wir uns auf ein strukturalistisch rekonstruiertes Beispielpaar. Wir vergleichen die historisch älteste Version mit der allgemeinsten Variante, also die Heider-Theorie HT mit der Holland-Leinhardt-Theorie HLT. Die möglichen intertheoretischen Relationen der anderen Theorien sollen anschließend betrachtet werden.

HLT ist im Vergleich zu HT die komplexere Theorie und behandelt mehr Anwendungen. Zunächst liegt es nahe, HT als *Spezialisierung* von HLT aufzufassen. Bei einer Spezialisierung muß jedoch der Begriffsapparat beider Theorien identisch bleiben, was hier nicht der Fall ist. Beispielsweise gibt es in HT zwei Relationen, in HLT hingegen eine Grundrelation und drei definierte Relationen. Es soll nun bewiesen werden, daß zwischen HT und HLT eine historische Reduktionsrelation vorliegt. Hierzu fassen wir die wichtigsten Definitionen noch einmal zusammen.

Potentielle Modelle x von HT bestehen aus einer (3-elementigen) Menge O von Objekten, einer Menge T von Zeitpunkten, der Kleiner-Relation $<$, sowie zweier Funktionen P: $T \rightarrow Pot(O \times O)$ und N: $T \rightarrow Pot(O \times O)$:
$x = \langle O, T, <, P, N \rangle$.
Das Axiom, welches in Modellen von HT gelten muß, drückt das Balanceprinzip aus:
(A1) Für alle $t \in T$ und für alle a: Wenn $t < max(T)$ und $a \in TR_{x(t)}$ und $a \in U_{x(t)}$, dann gibt es ein $t' \in T$, so daß gilt:
$t < t'$ und $a \in G_{x(t')}$ und für alle $t'' > t'$: $a \in G_{x(t'')}$
Die potentiellen Modelle x* von HLT bestehen aus einer endlichen Menge X von Objekten, einer Menge T von Zeitpunkten, der Kleiner-Relation $<$ und der Funktion R: $T \rightarrow Pot(X \times X)$. Potentielle Modelle von HLT haben also die Form:
$x^* = \langle X, T, <, R \rangle$.
Das Balancegesetz, welches in Modellen von HLT gelten muß, lautet:
(A2) Für alle $t, t' \in T$: wenn $t < t'$ dann $TRX(x^*_t) \leq TRX(x^*_{t'})$

Zur Definition der Reduktionsrelation ρ (D-2.1) sind die potentiellen Modelle und damit die Konzepte beider Theorien aufeinander zu beziehen. Die Begriffe müssen so ineinander überführt werden, daß intransitive Relationen aus HLT ungleichgewichtigen Heider-Triaden in HT entsprechen und (leer) transitive HLT-Relationen gleichgewichtigen Heider-Triaden. Es ist unmittelbar einsichtig, daß bei der Reduktion die jeweils ersten drei Tupelelemente von x und x* einander korrespondieren müssen. Die Zahl der Objekte X wird damit eingeschränkt auf $n = 3$. Weniger offensichtlich ist die Zuordnung der Relationen. In HT gibt es *zwei* Grundrelationen P und N, in HLT hingegen nur *eine* Relation R. Wenn wir jedoch die P-Relation von HT mit der M-Relation von HLT identifizieren und die N-Relation von HT mit der von HLT, so stehen wir mit dieser Deutung in Einklang mit der Behandlung der historisch älteren, graphentheoretisch ausgerichteten Theorien bei Holland/Leinhardt (1971): Holland

und Leinhardt setzen die positiven Relationen in den bewerteten Graphen der Vorgängertheorien mit M und die negativen Relationen mit N gleich (vgl. S.185).

Auf dem Hintergrund dieser Vorbemerkungen läßt sich die Reduktionsrelation $\rho \subseteq M_p(HLT) \times M_p(HT)$ wie folgt definieren.

(D-2.1') $\langle x^*,x\rangle \in \rho$ gdw
(1) $x^* = \langle X,T,<,R\rangle \in M_p(HLT)$ und
$x = \langle O,T',<',P,N'\rangle \in M_p(HT)$
(2) $X = O$, $T = T'$, $< = <'$, $N = N'$, $M = P$

Wir müssen nun die zentrale Forderung (2) aus Definition D-2 nachweisen, nach der unter ρ das Balancegesetz von HT aus dem von HLT folgt, oder modelltheoretisch: für alle x, x* gilt: wenn $x \rho x^*$ und $x^* \in M^*$, dann $x \in M$.
Zum Beweis benötigen wir einen einfachen Hilfssatz, der eine Beziehung zwischen den Heider-Triaden und dem Transitivitätsindex herstellt:

(Th-1) Ist $\langle x^*,x\rangle \in \rho$ dann gilt für alle $a \in TR$:
(1) $a \in U$ gdw $TRX(x^*) = 0$
(2) $a \in G$ oder $a \in I$ gdw $TRX(x^*) = 1$

Beweis von Th-1.1:
Sei $\langle x^*,x\rangle \in \rho$.
Angommen $\langle a,b,c\rangle \in U$. Dann ist nach Def. 3 von HT entweder $a \in N \wedge b,c \in M$ oder $b \in N \wedge a,c \in M$ oder $c \in N \wedge a,b \in M$. In der Definition des Transitivitätsindex (Def. 7 von HLT, S.191) wird der Ausdruck
$card\{\langle x,y,z\rangle |\ x,y,z \in X \wedge xRy \wedge yRz \wedge \neg xRz\} = 1$
und wegen n=3 $TRX(x^*) = 0$.
Angenommen umgekehrt $TRX(x^*) = 0$. Dann ist wegen $n = 3$
$card\{\langle x,y,z\rangle |\ x,y,z \in X \wedge xRy \wedge yRz \wedge \neg xRz\} = 1$,
d.h. es gibt eine Triade x,y,z mit $x,y,z \in X \wedge xRy \wedge yRz \wedge \neg xRz$. Da es unter ρ nur M- und N-Relationen gibt, muß gelten: $xMy \wedge yMz \wedge xNz$. Nach Def. 2.1' von ρ ist damit $\langle x,y\rangle$, $\langle y,z\rangle \in P$ und $\langle x,z\rangle \in N'$ und damit $\langle x,y,z\rangle \in U$.
Beweis von Th-1.2 analog □

Wir müssen nun unter Nutzung von (Th-1) zeigen, daß (A1) aus (A2) folgt. Eine einfache Überlegung zeigt, daß eine unmittelbare Deduktion nicht möglich ist. (A2) erlaubt nämlich, daß der Transitivitätsindex über die betrachtete Zeitperiode gleich bleibt, was für die Heider-Theorie die Konsequenz hätte, daß ein unbalancierter Zustand nicht verlassen werden muß. Das inhaltliche Axiom von HLT ist also zu schwach, um (A1) herleiten zu können und muß verschärft werden.

Eine naheliegende Spezialisierung von (A2) wäre die Verschärfung auf echte Ungleichheit der Transitivitätsindizes:
(A2') Für alle $t, t' \in T$: wenn $t < t'$ dann $TRX(x^*_t) < TRX(x^*_{t'})$
Ist das ursprüngliche Axiom zu schwach, so ist diese Spezialisierung viel zu stark und zu unrealistisch. Der Grund ist, daß unter (A2') für zwei beliebig dicht aufeinander-

folgende Zeitpunkte *immer* eine Transitivitätszunahme stattfinden müßte. Für die Herleitung von (A1) genügt die wesentlich schächere Annahme (A2"), daß es zwei Zeitpunkte geben muß, bei denen der zweite Index echt größer als der erste ist. Die den Spezialisierungen entsprechenden Modelle lassen sich in dem Mengendiagramm von Abb. 12.1 veranschaulichen.

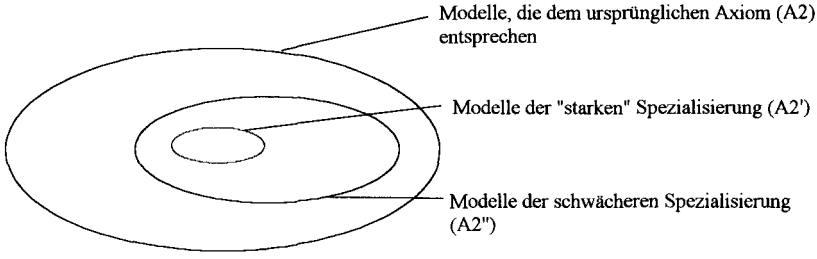


Abb. 12.1: Modellextension des ursprünglichen Axioms und zweier Spezialisierungen

Schließlich ist noch zu beachten, daß ein Transitivitätsindex 1 auch einer indefiniten Heider-Triade entspricht. Deren Herstellung muß jedoch ausgeschlossen werden, so daß sich als weitere Forderung ergibt, daß die Menge der M-Relationen nicht leer sein darf. Das ursprüngliche Axiom (A2) wird damit um zwei Bedingungen verschärft:

(A2") (1) Für alle t, t´ ∈ T: wenn t < t´ dann $TRX(x^*_t) \leq TRX(x^*_{t'})$ und
(2) Es gibt t, t' ∈ T: t < t´ und $TRX(x^*_t) < TRX(x^*_{t'})$ und
(3) Für alle t: $M_t \neq \varnothing$

Mit diesen Spezialisierungen können wir nun zeigen, daß (A1) aus (A2") formal folgt.
Angenommen x ρ x* und x* ∈ M*.
Angenommen weiter, t < max(T) und a ∈ $TR_{x(t)}$ und a ∈ $U_{x(t)}$
dann ist wegen Th-1.1 $TRX(x_t^*) = 0$
Nach (A2"-2) gibt es t' > t, so daß $TRX(x^*_{t'}) > TRX(x^*_t)$ und damit $TRX(x^*_{t'}) = 1$.
Also ist a ∈ $G_{x(t')}$ (da $M_t \neq \varnothing$ für alle t, ist a ∉ $I_{x(t')}$).
Sei weiter t" > t', dann (nach A2"-1) $TRX(x^*_{t''}) \geq TRX(x^*_{t'})$. Da $TRX(x^*_{t'}) = 1$ ist die Ungleichung erfüllt gdw $TRX(x^*_{t''})$ ebenfalls 1 ist (dies gilt für alle t" > t'). Damit ist a ∈ $G_{x(t'')}$ für alle t" > t'.
Gilt also x ρ x* und x* ∈ M* so folgt:
Wenn t < max(T) und a ∈ $TR_{x(t)}$ und a ∈ $U_{x(t)}$, dann gibt es ein t' ∈ T so daß gilt:
t < t' und a ∈ $G_{x(t')}$ und für alle t" > t': a ∈ $G_{x(t'')}$ □
Wir haben damit also gezeigt, daß die Heider-Theorie bei entsprechender Übersetzung und Spezialisierung auf die Holland-Leinhardt-Theorie formal reduzierbar ist. Transitivität ist damit tatsächlich das, was Heider Balance nennt. Jede Anwendung der Heider-Theorie kann durch die Reduktionsrelation in eine Anwendung der Holland-Leinhardt-Theorie übersetzt werden. HLT behandelt jene Fälle erfolgreich, die HT erfolgreich behandelt.

Auf die intertheoretischen Relationen der graphentheoretisch ausgerichteten Balancetheorien, die zwischen Heider und dem Transitivitätsmodell liegen, wurde in Kap. 9.2 bereits hingewiesen. Über Restriktionen der Triaden bzw. Kanten können diese als "Spezialfälle" bzw. Spezialisierungen (im vagen, nicht-strukturalistischen Sinn) von HLT betrachtet werden. Die Frage ist, welchen Relationen dies im strukturalistischen Wissenschaftskonzept entspricht. Obwohl die Theorien von Cartwright/Harary (1956), Davis (1967) und Davis/Leinhardt (1972) nicht strukturalistisch rekonstruiert wurden, läßt sich aus der informellen Darstellung erkennen, daß das Begriffsinventar von Cartwright/Harary und Davis relativ zu HLT unterschiedlich ist, das von Davis/Leinhardt hingegen identisch. Unter Verwendung des strukturalistischen Begriffsapparates dürften deshalb die Gleichgewichtstheorien von Cartwright/Harary (1956) und Davis (1967) in Reduktionsrelation zu HLT stehen, die Theorie von Davis/Leinhardt (1972) hingegen in Spezialisierungsrelation.[79]

12.2 Theorienevolution

Empirische Theorien können als Teil menschlicher Kulturgeschichte betrachtet werden, die wie jedes andere Kulturprodukt eine Geschichte haben: sie entstehen, entwickeln sich und sterben schließlich. Über die ganze Entwicklungsperiode bewahren sie dabei wie ein Staat oder eine Person eine Art "genidentische Identität" (Balzer/Moulines/Sneed 1987: 167). Eine solche Folge von historisch sich fortbildenden Theorie-Elementen oder Theoriennetzen bezeichnet man als *Theoriendynamik*, *diachrone* Theorienstruktur oder *Theorienevolution*.

Unter Theorienevolution wollen wir keine revolutionären Umwälzungen verstehen, sondern "normale Wissenschaft" im Sinne von Kuhn (1972). Normale Wissenschaft ist eine längere Phase in der Entwicklung von Theorien, in der die fundamentalen Konzepte und Theorieteile (das "Paradigma") nicht hinterfragt werden, die also im Rahmen einer bestimmten Tradition verrichtet wird. Dem Begriff der "normalen Wissenschaft" entspricht in etwa der des "wissenschaftlichen Forschungsprogramms" bei Lakatos (1982). Ein wissenschaftliches Forschungsprogramm im Sinn von Lakatos besteht aus einem "harten Kern", der durch eine umfangreiche "Schutzzone" von Hilfshypothesen gegen Widerlegung abgeschirmt ist. Ein leistungsfähiger Problemlösungsapparat "verdaut" dabei Anomalien und wandelt sie womöglich in positives Beweismaterial um. Lakatos (1982: 4) nennt als Beispiele für Forschungsprogramme die Newtonsche Gravitationstheorie, die Quantenmechanik, den Marxismus und den Freudianismus.

Die bisherige Darstellung der Balancetheorien läßt vermuten, daß diese ebenfalls als Forschungsprogramm oder ("normale") Theorienevolution interpretiert werden können. Über eine Periode von etwa 40 Jahren arbeiteten alle Gleichgewichtstheoretiker mit ähnlichen Konzepten und der grundlegenden Idee des "Strebens nach

---

[79] Dies gilt allerdings nur mit der Einschränkung, daß die Graphen in den Theorien von Cartwright/Harary (1956) und Davis (1967) vollständig sind. Vgl. hierzu S.233.

Gleichgewicht". Das Gleichgewichtsprinzip als fundamentaler Teil der Theorieentwicklung wurde dabei nicht hinterfragt und bildete die forschungsleitende Hypothese - den "harten Kern" - des balancetheoretischen Forschungsprogramms. Auftretende Anomalien widerlegten nicht das Gleichgewichtsprinzip, sondern wurden im Rahmen des Forschungsprogramms gelöst durch Einführung von Hilfshypothesen, Infragestellung des Anwendungsbereichs oder einfach Ignorierung (z.B. Markus/Zajonc 1985: 201: "However, many questions about balance remain unsolved, and some are perhaps not resolvable").

Die strukturalistische Theorienkonzeption bietet einen präzisierten Begriff der "Theorienevolution" bzw. des "Forschungsprogramms" (Stegmüller 1986: 107-127, Balzer/Moulines/Sneed 1987: 205-246). Zur genaueren Betrachtung der diachronischen Struktur von Gleichgewichtstheorien benötigen wir einige pragmatische bzw. sozio-historische Grundbegriffe. Wir halten uns bei der folgenden Darstellung an Kap. 5 aus Balzer/Moulines/Sneed (1987).

12.2.1 Konzepte zur Theorienevolution

In der bisherigen Verwendung des strukturalistischen Theorienapparates sind uns die "intendierten Anwendungen" als pragmatischer Begriff begegnet, der sich im Gegensatz zu M, M_p und M_{pp} nicht auf syntaktisch-semantische Art präzise definieren läßt. Bei der Betrachtung der Theorienevolution haben nun alle benötigten Grundbegriffe pragmatischen Charakter; im einzelnen sind dies die folgenden.

1. *Historische Perioden*
   Es wird angenommen, daß die Evolution einer Theorie in Perioden geteilt werden kann, die spezifische Merkmale aufweisen. Die historischen Perioden h_i konstituieren ein geordnetes Tupel $H = \langle h_1,...,h_i,...,h_n\rangle$, wobei h_1 den Anfang oder die "Geburt" der Theorie markiert und h_n den "Tod" oder die letzte betrachtete Periode. H ist dann die "Geschichte" der Theorie. Historische Perioden können in der Regel nicht numerisch exakt angegeben werden, sie sollten vielmehr als qualitative Fuzzy-Objekte betrachtet werden. Die Angabe von Jahreszahlen darf also nicht allzu genau genommen werden. Wichtiger ist zu sehen, daß einzelne Perioden auf andere mit unterschiedlicher Charakteristik folgen. In der Regel werden die einzelnen Perioden h_i auch nicht disjunkt sein und sich überlappen.
2. *Historische Präzedenz*
   Historische Präzedenz ist eine zweistellige, antisymmetrische und transitive Relation auf historischen Perioden. Damit soll ausgedrückt werden, daß eine bestimmte Periode h_i einer anderen Periode h_j historisch vorausgeht.
3. *Wissenschaftler*
   Ein Wissenschaftler soll hier als (potentieller) Nutzer einer Theorie betrachtet werden. Im Zeitalter der Künstlichen Intelligenz muß der Wissenschaftler nicht unbedingt menschlich sein. Vorstellbar sind auch bestimmte Klassen von Computern, die als Wissenschaftler fungieren können und Nutzer einer Theorie sind. Die Menge der Wissenschaftler wird mit SOPH bezeichnet.

4. Wissenschaftliche Gemeinschaft und wissenschaftliche Generation

Wissenschaftliche Gemeinschaften oder "Scientific Communities" (abgekürzt: SC) sind spezielle Teilmengen von SOPH. Eine SC wäre in unserem Fall die Gemeinschaft der Balancetheoretiker, also die Menge aller Wissenschaftler, welche mit dem Gleichgewichtsprinzip arbeiten. Diese Gemeinschaft der Balancetheoretiker kann als eine Wissensgemeinschaft oder genidentische Entität betrachtet werden. Damit ist gemeint, daß eine SC i.a. länger besteht als die aktive Lebensspanne irgendeines seiner Mitglieder dauert. Maßgebend für die Identität einer SC sind solche Dinge wie das gemeinsame Forschungsziel, gewisse von SC akzeptierte epistemische Standards sowie die fachliche Kooperation (Stegmüller 1986: 110). Gewöhnlich sind SCs keine hoch institutionalisierten Gruppen von Personen und die Grenzen sind im allgemeinen fuzzy.

Jede SC setzt sich wiederum aus (gewöhnlich mehreren) Teilmengen von SOPH zusammen, die wissenschaftliche Generationen (kurz: G) genannt werden. Die wissenschaftliche Gemeinschaft der Balancetheoretiker beispielsweise läßt sich in die erste Generation um Heider und in Folgegenerationen einteilen. Jede Generation G innerhalb einer SC ist eindeutig mit einer historischen Epoche assoziiert, in der die Gruppe wissenschaftlich aktiv ist. Formal kann dies mit einer Generationenfunktion g erfaßt werden. Ist HIST die Menge aller historischen Perioden und COMM die Menge aller SCs, so gilt:

$g: \text{HIST} \times \text{COMM} \rightarrow \text{Pot(SOPH)}.$

Jeder Wert $g(h_i,SC_j)$ ist eine wissenschaftliche Generation G_k.

Eine individuelle Person p kann natürlich sowohl unterschiedlichen Generationen als auch unterschiedlichen wissenschaftlichen Gemeinschaften angehören. Sie gehört zwei unterschiedlichen Generationen in derselben SC an, wenn ihr aktives wissenschaftliches Leben innerhalb von SC länger als eine Periode dauert. Analog gehört sie zwei verschiedenen Scientific Communities an, wenn sie gleichzeitig oder hintereinander völlig unterschiedliche Theorien benutzt.

Die Mitglieder einer Generation G kommunizieren untereinander in einer "spezifischen Wissenschaftssprache", die sich von der natürlichen Sprache mehr oder weniger scharf unterscheidet. Die Angehörigen von G teilen weiter besondere Techniken für das Beobachten, Klassifizieren und Systematisieren ihrer Studienobjekte. In den quantitativen Gebieten verfügen sie z.B. über spezifische Meßtechniken und -geräte sowie Berechnungsverfahren zum Testen von Hypothesen.

Eine wissenschaftliche Gemeinschaft ist sich in der Regel nicht ganz einig über die Anwendungen einer Theorie. Einige Anwendungen in I während einer Periode h werden von der SC als gut bestätigte Anwendungen eines Kerns K betrachtet werden, wir nennen diese F(I) (für "fester Anwendungsbereich"). Andere, weniger gut bestätigte Anwendungen werden vielleicht nur von einer Teilgruppe - im Extremfall nur einem Mitglied - von SC angenommen. Diese komplementäre Teilmenge von I soll A(I) (für "angenommener Anwendungsbereich") heißen. Dabei soll gelten: $F(I) \cup A(I) = I$ (und idealerweise, aber nicht notwendig: $F(I) \cap A(I) = \varnothing$). Ein Wechsel von Elementen aus A(I) nach F(I) kann dabei einen Wandel in der historischen Periode bedeuten. Der Begriff der "festen Anwendung"

ist dabei nicht absolut zu sehen, sondern relativ zu einer gegebenen Theorienevolution: bestimmte Anwendungen, die in einer Periode als fest angesehen werden, können später hinterfragt oder sogar aus den Anwendungen gestrichen werden. Die Ausdehnung der Menge der intendierten Anwendungen I wird - wie in Kap. 5.2 bereits erwähnt - als *empirischer Fortschritt* bezeichnet, die Ausdehnung der gesicherten Anwendungen F(I) als *epistemischer Fortschritt* (Stegmüller 1986: 111; 114).

Die pragmatischen Begriffe können nun in die Definition der Theorie bzw. des Theorie-Elementes T = ⟨K,I⟩ eingebaut werden. Das Ergebnis ist ein um pragmatische Elemente bereichertes diachrones Theorie-Element T' = ⟨K,I,G⟩. In diesem versucht eine wissenschaftliche Generation G in einer Periode h K auf I anzuwenden. Ein solches pragmatisch angereichertes Theorie-Element ist wie folgt definiert.

Definition
Wenn g eine Generationenfunktion ist, dann gilt:
T ist ein diachrones Theorie-Element gdw es K, I, SC, h, G gibt, so daß gilt:
(1) T = ⟨K,I,G⟩
(2) ⟨K,I⟩ ist ein Theorie-Element
(3) SC ist eine wissenschaftliche Gemeinschaft
(4) h ist eine historische Periode
(5) g(h,SC) =G
(6) G versucht K auf I anzuwenden

Jedes diachrone Theorie-Element kann als Momentfotografie des Wissenszustandes der wissenschaftlichen Generation G zur Zeit h verstanden werden. Die Evolution einer Theorie ist dann die historische Abfolge derartiger Momentfotografien. Der weitere formale Apparat, der hier nicht dargestellt werden kann, ist auf Theoriennetze und Spezialisierungen bezogen. Wir geben hier nur das Endprodukt - die strukturalistisch präzisierte Definition von Theorienevolution - wieder.

Definition
Eine Theorienevolution (im strukturalistisch präzisierten Sinn) ist eine endliche Folge $\langle N \rangle_i$ von Theoriennetzen, so daß für beliebige N_i, N_{i+1} in der Folge gilt:
(1) N_{i+1} folgt unmittelbar auf N_i
(2) Für alle $T_{i+1} \in |N_{i+1}|$ gibt es $T_i \in |N_i|$ so daß $T_{i+1}\ \sigma_d\ T_i$

N' folgt unmittelbar auf N, wenn es kein N_i gibt, das historisch zwischen N' und N liegt. σ_d in der zweiten Bedingung ist die diachrone Spezialisierungsrelation (für die genauen Definitionen vgl. Balzer/Moulines/Sneed 1987: 216-218). Die zweite Bedingung ist entscheidend für die Genidentität einer Theorie in einer Theorienevolution: Um von einer Theorienevolution im Sinn der Evolution von "ein und derselben Theorie" sprechen zu können, muß jedes Theorie-Element in einem neu konstruierten Theoriennetz über die Spezialisierungsrelation mit einem Theorie-Element in dem vorangegegangen Theoriennetz verbunden sein. Dabei ist es nicht

notwendig, daß alle intendierten Anwendungen im Verlauf der Theorienevolution Anwendungen gemeinsam haben. Vielmehr wird nur gefordert, daß es bei jedem Evolutionsschritt einige gemeinsame Beispielfälle gibt. Langfristig können die Anwendungen sich jedoch von den ursprünglichen Anwendungen stark unterscheiden, im Extremfall können sie auch vollständig verschieden sein.

Wir möchten den strukturalistischen Evolutionsbegriff etwas abändern, da er für unsere Zwecke zu streng erscheint. Zum ersten betrachten wir keine Theoriennetze, sondern Theorie-Elemente (dies ist eigentlich keine Modifikation, da sich dies auch so ausdrücken läßt, daß wir nur Netze mit genau einem Element berücksichtigen). Zum zweiten soll neben der Spezialisierungsrelation σ auch die Reduktionsrelation ρ die Genidentität einer Theorie ausdrücken dürfen.[80] Unter einer Theorienevolution verstehen wir damit eine Folge von n historisch aufeinanderfolgenden diachronen Theorie-Elementen $\langle T_1,...,T_i,...,T_{n-1},T_n\rangle$, wobei $T_1,...,T_i,...,T_{n-1}$ entweder in Reduktions- oder Spezialisierungsrelation zu T_n stehen. T_1 soll dabei die "Basis" heißen. Man beachte, daß nach dieser Definition das spezialisierte Theorie-Element zeitlich früher auftritt als das allgemeinere Theorie-Element. Dies stellt eine Umkehrung der allgemeinen Definition dar, trifft nach unserer Auffassung aber die Theorienentwicklung in den Sozialwissenschaften besser (vgl. die Bemerkung am Ende von Kap. 12.2.2, S.286-287). In dieser Definition drückt sich insofern auch ein *theoretischer Fortschritt* aus, als neuere Theorie-Elemente "allgemeiner" sind als ältere Elemente.

12.2.2 Evolution der Balancetheorien

Die grobe Entwicklung der Gleichgewichtstheorien wurde bereits angedeutet. Zur systematischen Darstellung bezeichnen wir diese Theorienevolution als "balancetheoretisch" - abgekürzt: B - und die wissenschaftliche Gemeinschaft als "Balancetheoretiker - kurz: SC(B).

B's Theorienevolution spielt sich von der Geburt bis zum vorläufigen Tod in einem überschaubaren Zeitrahmen von ca. 40 Jahren ab. Aus unserer Sicht sind wir berechtigt, vom "Tod" von B zu sprechen, als sich spätestens seit Mitte der achtziger Jahre keine entscheidenden theoretischen Verbesserungen ereignet haben. Witte (1989: 326) schätzt die derzeitige Forschungsaktivität als relativ gering ein und auch Opp (1984: 45) stellt fest, daß sich die Anzahl der Publikationen zu Balancetheorien deutlich verringert hat. Allerdings ist es ziemlich wahrscheinlich, daß B im Lauf der Wissenschaftsgeschichte weiterentwickelt wird, da es eine der fruchtbarsten theoretischen Strömungen in den Sozialwissenschaften darstellt (vgl. Stahlberg/Frey 1987: 219). Deshalb sprechen wir vom "vorläufigen" Tod.

Bevor wir auf das eigentliche Forschungsprogramm der Balancetheorien zu sprechen kommen, sei noch einmal an die bereits erwähnten balancetheoretischen Ideengeber und Nebenlinien erinnert. Das balancetheoretische Programm wurzelt in

---

[80] Für unsere Zwecke genügen Spezialisierung und Reduktion. Man kann sich aber vorstellen, daß weitere intertheoretische Relationen hinzugenommen werden müssen.

Spinozas Ethik, der Gestaltpsychologie von Wertheimer, Köhler und Koffka und in der Feldtheorie von Kurt Lewin. Der Schlüsselbegriff ist das gestaltpsychologische Prinzip der "kognitiven Konsistenz", nach dem Menschen bestrebt sind, ihr kognitives System widerspruchsfrei zu organisieren. Aus dem Konsistenzprinzip haben sich eine Reihe von Theorien ausdifferenziert, deren bekannteste neben der Balancetheorie die Dissonanztheorie von Festinger (1957) ist.
B läßt sich in vier historische Perioden einteilen, die sich z.T. stark überlappen.

Die *erste Periode* ist die Gründungsphase von B. Sie beginnt 1946 mit Heiders kurzem Artikel "Attitudes and Cognitive Organization", in dem der Theoriekern K(HT) einschließlich des Begriffsapparates und Balancegesetzes erstmals beschrieben ist.[81] Einen breiteren Raum nimmt die Darstellung seiner Theorie 1958 (deutsch: 1977) in "The Psychology of Interpersonal Relations" ein. In diesem Buch stellt Heider eine Fülle von Anwendungen vor, die als paradigmatischer Kern I_0 und fester Anwendungsbereich F(I) betrachtet werden können. Die oben berichteten Experimente von Jordan (1953), Lerner/Simmons (1966) oder Landy/Aronson (1969) stellen alle erfolgreiche Anwendungen des Theoriekerns dar. Bei der Benutzung des Theoriekerns stieß man allerdings auch auf Anomalien, wie z.B. die signifikanten Unterschiede innerhalb balancierter Triaden bei Jordan (1953). Diese initiierten eine länger anhaltende Diskussion und Folgeexperimente, die vereinzelt bis in die siebziger Jahre reichten. Heider zusammen mit den Anwendern seiner Theorie bildet die erste Generation von SC(B), die versucht haben, den Kern K(HT) auf I_0 anzuwenden.

In der *zweiten Periode* wurde der Formalismus und Anwendungsbereich von Heiders Theorie in verschiedene Richtungen ausdifferenziert und weiterentwickelt. Die bekanntesten Varianten sind: die interpersonell ausgerichtete Kommunikationstheorie von Newcomb (1953, 1961) mit der Einführung des Relevanzbegriffs (ein Objekt muß relevant für eine Person sein); die Kongruenztheorie von Osgood/Tannenbaum (1955), ebenfalls mit Anwendungen auf Kommunikationssituationen und die hier behandelte Theorie von Abelson/Rosenberg (1958). Das gemeinsame Merkmal dieser Phase ist eine Art *"theoretisches Experimentieren" mit dem Kern K(HT) unter minimalen Erweiterungen*. Von wenigen Ausnahmen abgesehen (z.B. Abelson/Rosenberg) bleiben die Theorien triadenbezogen. Variiert wird jedoch die Betrachtung der Qualität bestehender P- und N-Beziehungen. Die Varianten reichen von der Einführung neuer Relationstypen (O bei Abelson/Rosenberg, Relevanzrelation bei Newcomb) bis zu quantifizierenden Relationsbewertungen bei Osgood/Tannenbaum oder Mohazab/Feger (1985). Unter den intendierten Anwendungen finden sich neben Individualsystemen (kognitive Einstellungen) auch 2-Personen-Mikrosysteme mit sozialen Relationen (z.B. bei Newcomb). Die fünfziger Jahre bildeten den Fokus dieser Entwicklung, vereinzelt finden sich aber auch noch sehr viel später Ansätze, die

[81] Genau genommen existieren zu Heider (1946) zwei balancetheoretische Vorarbeiten aus dem Jahr 1944. Aber erst das dritte Papier von 1946 "enthält die explizite Formulierung der Balancehypothese" (Heider 1977: 13). In dem letzten Artikel von 1977 berichtet Heider noch einmal rückblickend von den Strömungen, die ihn zur Entwicklung der Balancetheorie bewogen.

dieser Phase zugerechnet werden müssen (z.B. Gollob 1974 oder Mohazab/Feger 1985). Typisches Kennzeichen all dieser theoretischen Weiterentwicklungen ist, daß diese - im Gegensatz zur nächsten Periode - relativ "unverbunden nebeneinander stehen" (Stahlberg/Frey 1987: 57), so daß eine Zuordnung zu einer einzigen Periode etwas fragwürdig bleibt. Eventuell können die Theorien dieser Periode in weitere Gruppen mit spezifischen Eigenschaften aufgeteilt werden.

Die *dritte Phase* markiert einen entscheidenden Bruch zu den beiden Vorgängerperioden. Anders als bei den Vorläufern können die Grenzen der historischen Periode hier exakt gezogen werden: sie beginnt 1956 mit der Arbeit von Cartwright und Harary und endet 1972 mit der von Davis und Leinhardt. Der Wandel wird durch zwei klare Charakteristika angezeigt. Das erste Merkmal ist die Verwendung der mathematischen Graphentheorie als präzisierter Sprache des Balancetheoretikers. Das zweite Merkmal ist die konsequente Erweiterung der potentiellen Anwendungsmöglichkeiten auf nicht-individuelle Phänomene mit dem Anwendungskern "objektiv beobachtbare Sozialrelationen". Die neue graphentheoretische Untermauerung des Balancemodells ist für Soziologen zudem eher zugänglich, "one may argue that it 'feels' more sociological" (Davis, nach Schenk 1984: 143). Die Arbeit von Cartwright und Harary bildet damit die "balancetheoretische Schnittstelle" zwischen psychologischen und soziologischen Anwendungen.

Inhaltlich ist diese Phase gekennzeichnet durch die Erforschung der *Konsequenzen* von Balance. Der Entwicklung in dieser Periode unterliegt eine fortschreitende Differenzierung der Folgen von Balancebedingungen: Von der Polarisierung in zwei Gruppen bei Cartwright/Harary (1956) über das Konzept der multiplen Gruppierung (Clustering) von Davis (1967) bis zur Hierarchisierung/Gruppierung von Davis/Leinhardt (1972). In den späteren Abschnitten bei Davis, Holland und Leinhardt ist insbesondere der Einfluß von Homans (1950) spürbar.

Als fester Anwendungsbereich F(I) der Balancetheorie in der dritten Periode werden vor allem informelle Gruppen wie Freundschaftsnetzwerke betrachtet. Wie in Kap. 9.3 gezeigt, wurde von einer Teilmenge von SC (Cartwright/Harary etc.) ein weiterer, angenommener Anwendungsbereich A(I) von formaleren Relationen, wie Beziehungen zwischen Gruppen, Organisationen, etc. diskutiert. Bei diesen Beispielen handelt es sich jedoch um nicht bestätigte, sondern nur vermutete bzw. diskutierte Anwendungen. Viele Argumente von anderen Balancetheoretikern (Hallinan, Felmlee, Anderson etc.) sprechen dagegen, daß diese Elemente der "hypothetischen Hoffnung" (Stegmüller 1986: 111) aus A(I) nach F(I) wechseln werden.

Die *vierte und letzte Periode* besteht aus unserer Sicht in der Entdeckung des Transitivitätsprinzips und der Einsicht, daß alle früheren Modelle Spezialisierungen bzw. Reduktionen dieser Theorie sind. Sie ist im wesentlichen mit der Arbeit von Holland und Leinhardt 1971 verknüpft, man kann aber auch die Variante von Johnsen (1985) noch dazunehmen. Allerdings erscheint die Abtrennung dieser von der letzten Periode eher künstlich, da sich sowohl die Wissenschaftler als auch die Konzepte stark überschneiden. Auch die festen und angenommenen Anwendungen F(I) und A(I)

bleiben dieselben wie in der dritten Periode. Wie die obengenannte Untersuchung von Davis (1970), Hallinan (1974) und Holland/Leinhardt (1975) zeigen, ist es gelungen, den Kern von HLT auf Freundschaftsnetze erfolgreich anzuwenden, so daß diese sich als gesicherte Anwendungen weiter bestätigt haben. Typisch in dieser und auch bereits gegen Ende der letzten Periode ist der Einsatz ausgefeilterer methodischer Verfahren zum Testen der Transitivitäts- bzw. Balanceannahme (z.B. Holland/Leinhardt 1970).

Nach dieser informellen Zusammenfassung dürfte die formale Darstellung der Theorieentwicklung mit dem strukturalistischen Begriffsinventar klar sein. Die Heider-Theorie ist die "Basis" der Theorienentwicklung. Die verschiedenen Personengruppen in jeder historischen Periode repräsentieren die "harten Kerne" erfolgreicher wissenschaftlicher Generationen, in welche die wissenschaftliche Gemeinschaft der Balancetheoretiker während der betrachteten Geschichte geteilt werden kann. Die historischen Perioden sind sehr fuzzy und überschneiden sich stark.

SC(B) = {"Balancetheoretiker"}

Erste Periode: h_1 = {1946- 196?}
G_1 = {Heider, Jordan, Lerner, Simmons, Landy, Aronson, ...}
I_1= $F(I_1)$ = {Individualsysteme kognitiv repräsentierter Triaden mit positiven und negativen sozialen Relationen}

Zweite Periode: h_2 = {1953-198?}
G_2 = {Newcomb, Osgood, Tannenbaum, Abelson, Rosenberg, Gollob, Mohazab, Feger, ...}
I_2 =$F(I_2)$ = $F(I_1) \cup$ {Individual- und 2-Personen-Systeme mit differenzierteren Beziehungen}

Dritte Periode: h_3 = {1956-1972}
G_3 = {Cartwright, Harary, Davis, Holland, Leinhardt}
I_3 =$F(I_3) \cup A(I_3)$
wobei
$F(I_3)$ = $F(I_1)$ ($\cup$ $F(I_2)$?) $\cup$ {Informelle Gruppen mit interpersonellen (Gefühls-) Beziehungen}
$A(I_3)$ = {Personen, Gruppen, Institutionen, Nationen mit *formalen* Beziehungen zwischen Elementen z.B. Machtsysteme, Kommunikationssysteme etc.}

Vierte Periode: h_4 = {1971-1985?}
G_4 = {Holland, Leinhardt, Davis, Hallinan, Felmlee, Johnsen, Hummell, Sodeur}
$I_4 = I_3$

Die eben gegebene Charakterisierung der Evolution von Balancetheorien läßt den Schluß zu, daß eine eindeutige, zeitlich lineare Abfolge von Gruppen und Theorie-Elementen, die jeweils auf Elementen von Vorgängerperioden aufbauen, nicht

existiert. Legt man den oben präzisierten Begriff der Theorienevolution zugrunde, kann man nicht von einer Theorienevolution im strengen Sinn sprechen. Die zweite "Periode" läuft z.B. zeitlich parallel zur dritten "Periode". Obwohl G_3 ein ausgefeiltes mathematisches Instrumentarium ausarbeitet, wird es von der praktisch parallel arbeitenden Generation G_2 nicht eingesetzt.

Die mehr oder weniger nebeneinander ablaufenden balancetheoretischen Strömungen sind ein bemerkenswertes wissenschaftstheoretisches Phänomen. Während naturwissenschaftliche Theorienevolution in der Regel so stattfindet, daß Theorie-Elemente zeitlich linear folgen und aufeinander aufbauen, kann bei den Balancetheorien davon keine Rede sein. Vielmehr scheint uns diese Situation ein typisches Merkmal der theoretisch unstrukturierten und empirisch bislang wenig erfolgreichen Sozialwissenschaften zu sein. Wir vermuten, daß mit Fortschreiten der Theorieentwicklung sich aber ähnliche Evolutionsmuster wie in den Naturwissenschaften ergeben und eine stärkere theoretische Geschlossenheit entstehen wird.

Es drängt sich die Interpretation auf, daß sich das Balanceprinzip von Heider in zwei "Evolutionslinien" ausdifferenziert: eine (sozial)psychologische Linie h_1 - h_2 und eine soziometrisch/soziologische Linie h_1 - h_3 - h_4. Diese Deutung läßt sich durch drei - zum Teil bereits genannte - Belege stützen. Erstens behandelt G_1/G_2 vorwiegend kognitive Individualsysteme, während bei G_3/G_4 die Anwendung auf "objektive" soziale Relationen im Vordergrund steht. Zweitens setzt sich G_1/G_2 aus Wissenschaftlern zusammen, die ausschließlich der Psychologie zugerechnet werden und deren Forschungsarbeiten ursprünglich in psychologischen Fachzeitschriften erschienen. Die Menge G_3/G_4 besteht hingegen aus mathematischen Soziologen und Statistikern, deren Arbeiten in soziologischen Zeitschriften veröffentlicht wurden. Eine Ausnahme ist der Aufsatz von Cartwright/Harary (1956), aber dieser kann als Bindeglied zwischen beiden Strömungen aufgefaßt werden. Ein dritter Beleg schließlich ist, daß in der soziologischen Literatur aus h_3 und h_4 zwar Heider als Bezugsquelle genannt wird, die anderen Aufsätze aus h_2 aber nicht erwähnt werden. Umgekehrt nehmen die psychologischen Balancetheoretiker G_2 die Arbeiten aus der Forschergeneration G_3/G_4 mit Ausnahme von Cartwright/Harary nicht wahr. Die wissenschaftliche Gemeinschaft der Balancetheoretiker verkettet dann zwar das Arbeiten mit dem Gleichgewichtsprinzip, sie zerfallen aber in die zwei Teilmengen der psychologischen und soziologischen Balancetheoretiker:

SC(B)={psychologische Balancetheoretiker} $\cup$ {soziologische Balancetheoretiker}[82]

Abb. 12.2 gibt auf der Basis der eben gemachten Überlegungen einen groben Überblick über die Entwicklung der Balancetheorien. Dabei wurden nur die wichtigsten Arbeiten eingetragen. Insbesondere in der sozialpsychologischen Entwicklungslinie findet sich eine Vielzahl von Theorien, die hier nicht aufgeführt werden konnten (vgl. z.B. die Literaturhinweise in Stephan 1990: 65). Zu bedenken ist auch, daß in der soziologischen Entwicklungslinie zwar Freundschaftsnetzwerke die präferierten Anwendungen sind, aber kognitive Systeme weiter in der Anwendungsmenge enthalten

[82] Das Schema stellt eine ganz nützliche, aber auch sehr grobe Vereinfachung dar. Soziometrische Gruppenstrukturen interessieren natürlich manche Sozialpsychologen und in Holland/Leinhardt (1975) finden sich z.B. auch Beiträge von Psychologen zu der soziologischen Evolutionslinie.

sind. Diese alten Anwendungen werden aber von diesen Generationen als "relativ uninteressant" in der Regel abgelehnt.

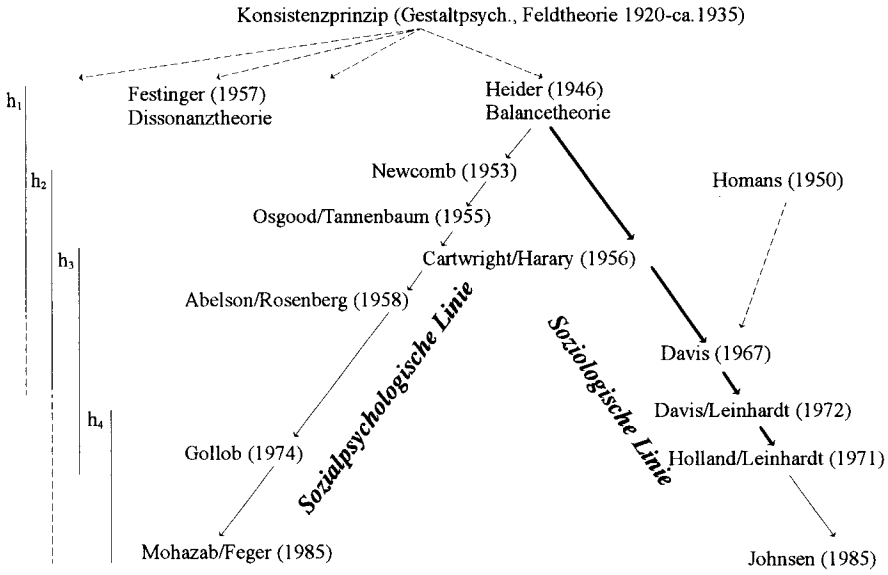


Abb.12.2: Evolution der Balancetheorien. Gestrichelte Pfeile bedeuten "Einflußfaktoren", durchgezogene, dünne Pfeile Evolution im schwachen Sinn und dicke Pfeile Evolution im oben präzisierten Sinn. Die Folge der durch dicke Pfeile verbundenen Theorie-Elemente steht in Reduktions- oder Spezialisierungsrelation zum letzten Element.

Die Theorienfolge in der soziologischen Linie

- Heider (1946, Basis)
- Cartwright/Harary (1956)
- Davis (1967)
- Davis/Leinhardt (1972)
- Holland/Leinhardt (1971)

ist eine Theorienevolution im erwähnten und präzisierten Sinn. Die ersten vier Elemente sind Spezialisierungen oder Reduktionen der Holland-Leinhardt-Theorie. Daß das allgemeinere Modell Holland/Leinhardt (1971) vor dem spezielleren Davis/ Leinhardt (1972) publiziert wurde, hat wohl veröffentlichungstechnische Ursachen. Den wissenschaftstheoretischen Status von Johnsen (1985) lassen wir hier außer Betracht.

Die Perioden h_3 und h_4 in der soziologischen Linie stellen außerdem einen empirischen Fortschritt im Vergleich zu h_1 dar, da sich die Menge der intendierten Anwendungen (wenn auch nicht sehr stark) vergrößert hat. h_4 ist ein epistemischer Fortschritt, weil sich die Menge der gesicherten Anwendungen von HLT erweitert hat. Außerdem kann h_4 als theoretischer Fortschritt im Vergleich zu den Vorgängerperioden in der soziologischen Linie betrachtet werden, da sich die anderen Theorie-

Elemente in dieser Linie auf HLT reduzieren lassen bzw. Spezialisierungen sind. Analog dürfte h_3 ein theoretischer Fortschritt im Vergleich zu h_1 sein, da sich die Heider-Theorie auf die Cartwright-Harary-Theorie zurückführen lassen müßte. Eine Theorienevolution, die für jeden Zeitabschnitt h entweder einen empirischen oder theoretischen oder epistemischen Fortschritt darstellt, kann als *fortschrittliche Theorienevolution* oder *fortschrittliches Forschungsprogramm* im Sinn von Lakatos (1982) bezeichnet werden (vgl. zu diesen Begriffen Stegmüller 1986: 114-115).[83] Die soziologische Evolution der Balancetheorien (h_1 - h_3 - h_4) kann also als fortschrittliches Forschungsprogramm im Sinn von Lakatos (1982) aufgefaßt werden.

In der sozialpsychologischen Entwicklungslinie können wir unter Zugrundelegung der obengenannten Definition vermutlich nicht von Theorienevolution sprechen. Obwohl dies genauer untersucht werden müßte, läßt sich doch der informellen Literatur entnehmen, daß die Theoriekerne relativ unverbunden nebeneinander liegen (Stahlberg/Frey 1987: 57).[84] Man kann die historische Entwicklung der Balancetheorien dann allenfalls als Evolution in einem sehr schwachen Sinn auffassen, bei der die einzelnen Theorie-Elemente nur insofern "verbunden" sind, als sie das Heidersche Gleichgewichtsprinzip "in irgendeiner Weise" variieren.[85] Möglicherweie können bestimmte Theorie-Elemente bzw. Gruppen durch Spezialisierung oder Reduktion verknüpft werden, mit großer Wahrscheinlichkeit aber nicht alle Elemente. Ein erster Anfang ist die Arbeit von Kuokkanen (1992), der eine Teilmenge von Balancetheorien in der sozialpsychologischen Linie im strukturalistischen Format rekonstruiert. Kuokkanen kommt zu dem Ergebnis, daß es keinen einfachen kumulativen Fortschritt bei diesen Theorien gibt und ein Teil dieses Theorien inkompatibel zueinander ist (z.B. die Theorie von Heider und Osgood/Tannenbaum).

Die herausgearbeitete fortschrittliche Entwicklung der Balancetheorien in der soziologischen Linie stützt eine Hypothese von Stephan (1990: 147), die sich auf der Basis der Rekonstruktion mehrerer psychologischer Theorien herausschälte. Nach dieser Hypothese besteht ein systematischer Unterschied zwischen physikalischen und sozialwissenschaftlichen Theorien dahingehend, daß physikalische Theorien durch *Spezialisierungen*, sozialwissenschaftliche Theorien dagegen durch *Erweiterung bzw. Differenzierung* ihres Begriffsapparats zu gehaltvolleren Aussagen kommen. Dies bestätigt sich in unseren Theoriebeispielen zumindest tendenziell: HT verwendet nur

---

[83] Die Begriffe "empirischer Fortschritt", "theoretischer Fortschritt" und "fortschrittliche Theorienevolution" werden hier in einem laxeren Sinn verwendet als bei Stegmüller (1986: 114-115).

[84] Wiswede (1988: 17) hat den Ausdruck geprägt, daß sozialpsychologische Theorien keine "Baumstruktur", sondern eine "Spaghettistruktur" haben, und dies dürfte in weiten Teilen des balancetheoretischen Programms (wie der meisten sozialwissenschaftlichen Theorien) zutreffen.

[85] In diesem schwachen Sinn rechtfertigt sich dann auch die Redeweise vom "Paradigma" der Balancetheorien. In der strukturalistischen Deutung des Paradigmabegriffs von Kuhn wird das historisch erste Theorieelement $\langle K_0, I_0 \rangle$ als "Paradigma" identifiziert (für eine differenziertere Darstellung vgl. Stegmüller 1986: 115-117). Heider (HT) hat den allgemeinen theoretischen Rahmen K_0 (die mathematische Grundstruktur) geschaffen und eine Menge von Anwendungsfällen I_0 genannt. Im weiteren Verlauf der Entwicklung wurde der Theoriekern variiert bzw. erweitert und neue Anwendungen gefunden. Die in der balancetheoretischen Tradition arbeitenden Normalwissenschaftler gaben zu keiner Zeit das von Heider formulierte Balanceprinzip auf oder sahen es gar als widerlegt an.

drei Objekte, ART und die Theorien in der soziologischen Linie unbegrenzt viele; HT benutzt zwei Relationen, die nach und nach (wenn auch nicht durchgehend) differenziert werden (M-, A-, N-Relationen in HLT). Die Differenzierung betrifft aber nicht nur den eigentlichen Begriffsapparat, sondern den ganzen Theoriekern: in den Naturwissenschaften wird zuerst das *allgemeine Theorie-Element* eingeführt und dann sukzessive spezialisiert (vgl. die Beispiele in Balzer/Moulines/Sneed 1986), in den Sozialwissenschaften beginnt man hingegen anscheinend mit einem *einfachen, sehr speziellen Theorie-Element* (HT) und führt nach und nach Generalisierungen durch (Cartwright/Harary - Davis - HLT). Es scheint, daß die Erweiterung und Differenzierung von Theoriekernen eine Eigenart sozialwissenschaftlicher Theoriebildung ist.

13. Zusammenfassung und Schlußfolgerungen

Das primäre Ziel dieser Arbeit war, die Möglichkeiten und Grenzen eines KI-basierten Modellierungsansatzes in Verbindung mit einer neuen Betrachtungsweise von Theorien an einem konkreten Beispiel aufzuzeigen. Bei dem KI-Modellierungsansatz handelte es sich um wissensbasierte Systeme, also Computerprogramme, die vorrangig deklaratives Wissen statt numerischer Algorithmen enthalten. Die neue Betrachtungsweise von Theorien basierte auf dem strukturalistischen Theorienkonzept der analytischen Wissenschaftstheorie, das bislang in den Sozialwissenschaften kaum angewendet wurde. Das konkrete Beispiel bildeten Gleichgewichtstheorien.

Teil I: Grundlagen

Ausgangspunkt war die Rekonstruktion von Theorie- und Modellbegriffen, wie sie in den empirischen Sozialwissenschaften und in der Simulationsliteratur verwendet werden. Zunächst wurde das Standardtheorienkonzept des Logischen Empirismus und Kritischen Rationalismus - in der Rezeption von Sozialwissenschaftlern - als metatheoretischer Rahmen gewählt. Aus der Simulationsliteratur ließen sich unterschiedliche Modellbegriffe ableiten, und es mußte festgestellt werden, daß die Beziehung von "Modell" und "Theorie" mehrdeutig ist, da der Modellbegriff im Standardtheorienkonzept nicht explizit Teil der Theoriedefinition ist.

1. Computermodellierung

Es folgten einige grundlegende Betrachtungen über die Modellierung theoretischer Phänomene mit dem Computer und Anwendungsmöglichkeiten in den Sozialwissenschaften. Der Vorteil des Computers bei der Theoriebildung liegt vor allem im Formalisierungszwang, der maschinellen "Ausführbarkeit", der unumschränkten Ableitungsfähigkeit und im experimentellen Charakter. Die gegenwärtig in den Sozialwissenschaften hauptsächlich verwendeten maschinellen Modellierungsformen sind weitgehend quantitativ orientiert und oft sogar an dem physikalistischen Ideal der Formulierung von Differentialgleichungen. Dies kontrastiert zum faktischen Stand sozialwissenschaftlicher Theorienbildung, bei dem Theorien und Gesetzmäßigkeiten meist nur qualitativ vorliegen und trotzdem z.T. bedeutende theoretische Einsichten liefern. Unabhängig davon ist die konventionelle Modellierung dadurch gekennzeichnet, daß die Entstehungsmechanismen der erzeugten Outputs unklar sind und theoretisches Wissen mit Steuerungsprozeduren vermischt wird. Als Ausweg wurde für eine qualitative Modellierung auf der Basis von wissensbasierten Systemen argumentiert. Dieser KI-basierte Ansatz reduziert bzw. beseitigt die mit der konventionellen Modellierung verknüpften Nachteile.

Anhand von theorie- und datengesteuerten Modellierungsansätzen wurde die in der Simulationsliteratur oft geäußerte Auffassung untersucht, nach der Computerpro-

gramme unmittelbar Theorien darstellen. Im theoriegesteuerten Vorgehen übersetzt man eine explizit vorliegende Theorie in ein Computerprogramm. Unterlegt man das Standardtheorienkonzept, so kann das Programm als Repräsentation der Theorie aufgefaßt werden, wenn sie die wesentlichen theoretischen Aussagen - das sind insbesondere die postulierten Gesetzmäßigkeiten - enthält. Im Gegensatz zum theoriegesteuerten Vorgehen geht man bei der datengesteuerten Methode nicht von Theorien aus, sondern von Daten und versucht in einem Generate-and-Test-Verfahren Programme zu schreiben, die ein möglichst optimales Input/Output-äquivalentes Verhalten zum empirischen System zeigen. Gelingt dies, so wird das Programm oft als Theorie über das abgebildete Phänomen betrachtet. Diese Sicht- und Vorgehensweise ist typisch für die Kognitionswissenschaften. Nach Auffassung vieler Kognitionstheoretiker erklären die Programme vollständig die modellierten kognitiven Phänomene, wenn sie in der Lage sind, diese Input/Output-äquivalent zu generieren. Der Theorieperspektive der Kognitionswissenschaftler unterliegt damit ein kausalmechanistisches Erklärungsmodell, welches grundlegend vom Subsumtions-Modell des Standardtheorienkonzepts abweicht.

2. Wissensverarbeitung

Im zweiten Grundlagenteil wurden die elementaren Konzepte der Wissensverarbeitung vorgestellt, wie sie in der Künstlichen Intelligenz (KI) ausgearbeitet wurden. Wissensbasierte Systeme sind Computerprogramme, die das Wissen eines Fachgebiets auf dem Rechner verfügbar machen. Der Computer repräsentiert und verarbeitet das Wissen, zieht Schlußfolgerungen und erklärt die deduzierten Schlüsse. Wissensbasierte Computerprogramme in der Architektur von Expertensystemen ermöglichen eine modulare, transparente und flexible Modellierung von Wissen - wie es auch in Theorien enthalten ist - in Form von Fakten und Regeln. Regeln können Definitionen und Gesetze einer Theorie repräsentieren, die auf empirische Daten (Fakten) angewendet werden und so neue Schlußfolgerungen produzieren. Ein wesentlicher Vorteil solcher wissensbasierter Modelle ist die deduktive Mächtigkeit, die nichtalgorithmische, symbolische Darstellungsmöglichkeit theoretischer Konzepte und die Existenz einer klaren Schnittstelle zwischen eigentlichem Wissen und Prozeduren zur Bearbeitung des Wissens. Für die Modellierung qualitativer Theorien genügen Prädikatenlogik und Mengenlehre und damit auf maschineller Seite Programmiersprachen, die Konzepte von Logik und Mengentheorie ausdrücken können. Eine solche Sprache ist PROLOG. In Verbindung mit Expertensystem-Techniken erlaubt PROLOG eine natürliche, deklarative Repräsentation qualitativer Theorien, eine klare Trennung von Regeln - und damit theoretischer Teile - von der Abarbeitung der Regeln, eine Erklärung abgeleiteter Schlüsse und einen modularen und flexiblen Modellaufbau. In den Sozialwissenschaften wurde das Modellierungspotential dieses Ansatzes bereits erkannt, insbesondere für die Computermodellierung bislang mit konventionellen Instrumenten nicht formalisierbarer - auch klassischer - soziologischer Theorien.

3. Wissenschaftstheorie

Im dritten Grundlagenteil wurde dafür plädiert, Theorien nicht naiv in Programme zu übersetzen, sondern vor der Übersetzung den logischen Aufbau explizit zu machen. Als metatheoretische Theorie, die diesen logischen Aufbau expliziert, wurde ein neuerer Ansatz der analytischen Wissenschaftstheorie verwendet: das strukturalistische Theorienkonzept. In diesem Konzept werden Theorien nicht mehr als Satzmengen interpretiert wie im bislang unterlegten Standardtheorienkonzept. Vielmehr bestehen Theorien in dieser Deutung aus einem mathematischen Strukturkern, der u.a. das Fundamentalgesetz oder "eigentlich inhaltliche Axiom" enthält und einer Menge von Beispielen, auf das der Strukturkern angewendet wird. In diesem Konzept gibt es insbesondere eine klar definierte Relation zwischen "Theorie" und "Modell" in dem Sinn, daß Modelle die Extensionen (Beispielfälle) der Axiome einer Theorie sind. Wissensbasierte KI-Modellierungstechniken und die strukturalistische Wissenschaftstheorie können beide als metatheoretische Ansätze aufgefaßt werden, die wesentliche Gemeinsamkeiten aufweisen: sie beschäftigen sich mit Formalismen zur Repräsentation von Wissen und sie brechen mit bestimmten physikalistisch-positivistischen Traditionen. Die strukturalistische Wissenschaftstheorie verwirft das mit dem Standardtheorienkonzept und insbesondere dem Logischen Empirismus verknüpfte formalsprachliche, normative und physikalistisch orientierte Vorgehen. Der wissensbasierte Modellierungsansatz löst sich vom Zwang der konventionellen, quantitativ-algorithmisch orientierten Computermodellierung und erlaubt sogar die maschinelle Repräsentation von vagem "Fuzzy"-Wissen. Beide Konzepte sind also wesentlich liberalere metatheoretische Formalismen.

Teil II: Balancetheorien

Im zweiten Teil der Arbeit wurde versucht, beide Meta-Programme auf konkrete Theorien aus den Sozialwissenschaften anzuwenden. Betrachtet wurde eine Folge von Gleichgewichtstheorien. Diese liegen zwar z.T. mathematisch formalisiert vor, aber weder existiert eine Betrachtung des Theorienstrangs in einem einheitlichen metatheoretischen Rahmen, noch eine Computermodellierung auf symbolischer Basis. Für den hier vertretenen Ansatz sind Gleichgewichtstheorien prädestinierte Kandidaten, da es primär um Strukturen und nicht um Quantitäten geht.

Ausgangspunkt waren die kognitiv ausgerichteten Theorien von Heider (HT, 1946) und Abelson/Rosenberg (ART, 1958). Deren zentrale Behauptung bestand darin, daß triadische (HT) oder größere kognitive Strukturen (ART) zu Gleichgewicht tendieren und in gewisser Weise verändert werden. Cartwright/Harary (1956) wendeten die graphentheoretisch verallgemeinerte Heider-Theorie auf soziale Strukturen an und zeigten, daß Balance zu einer dichotomen Gruppenbildung führt. Davis (1967) entwickelte das Modell weiter zum Clusterabilty-Modell, welches mehr als zwei Gruppen erlaubt, Davis/Leinhardt (1972) führten im Ranked-Clusters-Modell zusätzlich Hierarchien ein, und Holland/Leinhardt (HLT, 1971) entwickelten das T-Graph-Modell.

Unter Annahme der Transitivitätstendenz konnte in HLT gezeigt werden, daß mit Balancetendenz eine Gruppenstruktur aus hierarchisch geordneten Cliquen entsteht.

1. Wissenschaftstheoretische Ergebnisse (logische Struktur)

Unter wissenschaftstheoretischen Gesichtspunkten sind Balancetheorien vor allem aus zwei Gründen interessant. Erstens bilden Gleichgewichtstheorien ein schönes Beispiel dafür, wie sich aus einem einfachen Prinzip mit sehr beschränkten Anwendungen ein fachübergreifendes theoretisches Forschungsparadigma entwickeln kann. Zweitens stellten sich Balancetheorien in der Vergangenheit insofern als theoretisch äußerst fruchtbar heraus, als es die späteren Varianten erlaubten, aus einfachen Annahmen über individuelle Eigenschaften von Akteuren auf der Mikroebene strukturelle Folgerungen für die Makroebene zu erzeugen.

In der logischen Rekonstruktion des verfolgten Theorienstrangs trat die verwendete Begrifflichkeit deutlich hervor. Alle Gleichgewichtstheorien arbeiteten mit einer endlichen Menge von Objekten und 2-stelligen Relationen auf diesen Mengen. Ein durchgehendes Charakteristikum aller Balancetheorien war das Fehlen theoretischer Terme. Diese Eigenschaft kann als Hinweis auf die konzeptuelle Einfachheit der Gleichgewichtstheorien im Vergleich etwa zu naturwissenschaftlichen Theorien gewertet werden. Ein weiteres Charakteristikum war, daß alle inhaltlichen Axiome oder Gesetze in den unterschiedlichen Versionen der Balancetheorien die gleiche einfache und grundlegende Idee ausdrückten: unbalancierte Strukturen werden tendenziell in balancierte Strukturen überführt. Mit einer Ausnahme (ART) ließen alle Theorien unbestimmt, nach welchen Regeln diese Überführung erfolgt. In der strukturalistischen Interpretation wurde bei der Heider- und Holland-Leinhardt-Theorie das Verhältnis von Theoriekern und empirischer Anwendung genauer herausgearbeitet. Die Diskussion um den Anwendungsbereich bestätigte auch für die Sozialwissenschaften die strukturalistische Sichtweise, daß Anwendungen nicht automatisch mit einer Theorie mitgeliefert werden. Da nicht ganz klar ist, auf welche empirischen Systeme die Theorien passen, empfiehlt der Strukturalismus, den Theoriekern selbst über die Anwendungen entscheiden zu lassen und über die Regel der Autodetermination zu einer Charakterisierung des Anwendungsbereichs zu kommen.

Der intertheoretische Vergleich der verschiedenenen Balancetheorien erbrachte interessante Erkenntnisse. Anders als in den Naturwissenschaften gibt es im balancetheoretischen Paradigma eine Vielzahl von Variationen des Basiselements (HT). Diese Varianten wurden zeitlich teilweise parallel entwickelt, ohne daß sie sich immer jeweils aufeinander beziehen und sukzessive ausgearbeitet wurden. Dieser Sachverhalt demonstriert deutlich die mangelhafte theoretische Geschlossenheit der Sozialwissenschaften. Für eine Teilmenge der balancetheoretischen Theorie-Elemente konnte jedoch gezeigt bzw. erwartet werden, daß diese durch intertheoretische Relationen (Reduktions- und Spezialisierungrelationen) verbunden sind. Die Folge dieser Theorien bildet damit eine Theorienevolution und - aufgrund theoretischer und empirischer Erweiterungen - ein Forschungsprogramm im Sinn von Lakatos. Die verfolgte Entwicklungslinie von Balancetheorien verweist auf einen wichtigen

Unterschied sozialwissenschaftlicher Theorien im Vergleich zu naturwissenschaftlichen Theorien: während bei naturwissenschaftlichen Theorien theoretisch gehaltvollere Aussagen vor allem durch Spezialisierungen entstehen, scheint dies in den Sozialwissenschaften eher durch Erweiterung und Differenzierung des Theoriekerns zu erfolgen.

2. Wissensbasierte Modellierung

Drei Versionen dieses Theorienstrangs - HT, ART und HLT - wurden in einem wissensbasierten Ansatz in ein Computerprogramm übertragen. An allen drei Versionen wird deutlich, wie natürlich die Übertragung theoretischer Konzepte von der Quellsprache (informelle Darstellung und strukturalistische Interpretation) in die Zielsprache (PROLOG) ohne großen semantischen Bruch erfolgt. Weder müssen deklarative Teile algorithmisiert werden, noch müssen Schritte zum Lösungsablauf vorgeschrieben oder die Theorie prozessionalisiert werden. Theoretische Teile sind explizit codiert und deutlich von nicht-theoretischen Teilen getrennt, so daß z.B. Regeln hinzugefügt oder weggenommen werden können, ohne einen Eingriff in Steuerungsprozeduren vornehmen zu müssen. Ein wesentlicher Vorteil bei der Implementierung der HL-Theorie war, daß der dabei entwickelte Inferenz- und Erklärungsmechanismus unabhängig von der gegebenen Anwendung ist. Dieser Systemkern kann nun für beliebige andere Theorien weiterverwendet werden, so daß praktisch nur mehr die theoretischen Aussagen in die Regeln transformiert werden müssen. Wir haben dies nicht an einer Theorie, sondern an soziometrischen Definitionen demonstriert.
An allen implementierten Theorieversionen konnten *unterschiedliche* Aspekte und Vorteile der regelbasierten Modellbildung gezeigt werden.

1. Die einfache Heider-Theorie HT wird benutzt, um im Rahmen des strukturalistischen Theorienkonzepts die beiden grundlegenden Arten aufzuzeigen, Theorien in Programmen darzustellen. In der ersten Version wird HT als Datengenerierungs-Modell implementiert, das aus vorliegenden Daten unter Benutzung des "inhaltlichen Axioms" neue Daten erzeugt. In der zweiten Version werden direkt die strukturalistischen Prädikate zur Theorie implementiert, so daß diese auf eine gegebene Datenstruktur angewendet und überprüft werden können. Das Programm übernimmt die deklarative Rolle der Axiome und erzeugt "metaempirische" Daten. Beispielsweise ist es damit möglich, beliebige Datenbasen daraufhin zu untersuchen, ob diese Modelle oder potentielle Modelle von HT sind oder überhaupt nichts mit der Theorie zu tun haben.
2. Bei der Verallgemeinerung von HT in ART wird im Datengenerierungs-Modell eine z.T. große Menge von Schlußfolgerungen erzeugt, die teilweise inkonsistent im Sinn der Theorie sind. An diesem Beispiel zeigt sich der allgemeine Vorteil, daß die Maschine im Gegensatz zu "natürlichen Wissenschaftlern" vollständig und korrekt große Datenmengen ableiten kann. Eine einfache Erklärungskomponente ermöglicht es, die von den Axiomen der AR-Theorie erzeugten Schlüsse rechtfertigen zu lassen. Mit den abgeleiteten Daten in Verbindung mit der Erklärungskomponente ist es möglich zu prüfen, ob das Modell "vernünftige" Ergebnisse

produziert, und es können "Aha-Effekte" ausgelöst werden. Dies kann zur Revision und Modifikation der Theorie oder von Theorieteilen führen und damit die Weiterentwicklung von Theorien fördern. Es kann aber auch die Infragestellung der Theorie für bestimmte intendierte Anwendungen zur Folge haben. Im vorliegenden Fall eignet sich das Modell insbesondere für empirische Tests mit Versuchspersonen.

3. Die dritte implementierte Theorie von Holland-Leinhardt (HLT) ist wohl das interessanteste Modell, da es über die deduktiven und erklärenden Möglichkeiten des AR-Modells hinaus deutlich macht, wie durch Experimente theoretische Konsequenzen bestimmter Annahmen erforscht und entdeckt werden können. Eine einfache Einsicht, die unsere heuristischen Experimente gebracht haben war, daß die Behauptung, strukturelle Balance sei ein Spezialfall des transitiven Graphen, nur für vollständige Graphen gilt - was mit dem ursprünglichen Modell von Cartwright/Harary nicht vorausgesetzt war.

Wir wollen die wesentlichen Ergebnisse der Experimente mit der HL-Theorie noch einmal kurz zusammenfassen. Das zentrale inhaltliche Axiom der Theorie behauptet, daß empirische Strukturen, wie z.B. soziale Gruppen, eine Tendenz zu Transitivität und damit Balance besitzen und als Folge davon in horizontal und vertikal geschichtete Cliquen partitioniert werden können. Das Axiom sagt nichts darüber aus, *wie* eine Überführung in balancierte Strukturen erfolgt. In unser Modell wurden verschiedene Hypothesen eingeführt, wie diese Transformation erfolgen könnte, d.h. es wurden verschiedene Spezialisierungen des Fundamentalgesetzes der Theorie vorgenommen. Danach wurden die Annahmen im Modell simuliert und geprüft, inwieweit die Konsequenzen empirisch und theoretisch sinnvolle Ergebnisse produzieren. Zugrundegelegt wurde eine rein strukturelle Perspektive.

Die fundamentale theoretische Annahme war, daß bestehende Cliquen Initialisierungsträger für Balancetendenzen sind. Die Simulation zeigt, daß mit der Herstellung von Transitivität in bestimmter Weise miteinander verbundene Cliquen zusammenfallen, so daß u.U. die ganze Gruppe zu einer einzigen Clique kollidieren kann. Empirisch ist das Zusammenfallen von Cliquen selten zu beobachten, so daß wir das Balancegenerierungs-Verfahren unter der Randbedingung modifizierten, daß Cliquen möglichst nicht vereinigt werden. Einfache theoretische Überlegungen zeigen, daß dies nur möglich ist, wenn Relationen entfernt werden.

Wichtige theoretische Konsequenz des so implementierten Modells war, daß sowohl die Auflösung bestehender Cliquen als auch die Entstehung neuer Cliquen und die Integration von Nicht-Cliquenmitglieder in Cliquen erklärt werden kann. Mit Transitivitätsdruck wurden ausgehend von Cliquen mit höherem Status Verbindungen zu Cliquen mit niederem Status "gekappt", falls die Gefahr des Zusammenfallens existierte. Unmittelbar miteinander z.B. über Liaisonpersonen verknüpfte Cliquen wurden vom Herd der Clique mit höherem Status aus aufgelöst. Andererseits wurden über Wahl-Zyklen mit Cliquen verbundene Personen diesen einverleibt und aus zyklen-verknüpften nicht-cliqualen Personen entstand eine neue Clique. Inwieweit diese Modellkonsequenzen in der realen Welt zutreffen, muß empirischer Forschung überlassen werden.

Weitere interessante theoretische Konsequenzen ergaben sich, als wir unser Transitivitätsgenerierungs-Verfahren mit der Entwicklung schwacher und starker Beziehungen ("weak" und "strong ties") verknüpften. Transitivitätszunahme bedeutete zunächst, daß die intercliqualen "schwachen" Beziehungen zwischen Cliquen ansteigen und damit interessante neue Informationen über immer mehr Personen von einer Clique in die andere wandern. Waren die Cliquen hingegen beidseitig miteinander verknüpft, so bestand mit Transitivitätstendenz bei Cliquen mit höherem Status vermehrt die Neigung, intercliquale schwache Relationen zu der verknüpften Clique mit niedrigem Status abzubauen - also Informationsabgabe einzustellen - während umgekehrt die Clique mit niederem Status vermehrt schwache Relationen zur höheren einführt und diese immer mehr mit Informationen versorgt. Mit Balancetendenz erfolgt also eine Hierarchisierung von Wissen in dem Sinn, daß hierarchisch höher stehende Cliquen immer mehr Informationen aus niedrigeren erhalten, aber nicht umgekehrt.

Es sei zum Schluß noch auf zwei Schwachstellen der Regelmodellierung mit den vorliegenden Theorien hingewiesen. Als eine der wesentlichen Konsequenzen des hier vorgestellten Ansatzes wurde die erklärende und deklarative Modellierungsweise betont. Diese Vorzüge kamen bei den *vorliegenden* Theorien nicht so stark zur Geltung, wie sie bei realen Expertensystemen normalerweise zur Geltung kommen. Ein Grund ist, daß die theoretisch relevante Regelmenge bei den vorliegenden Theorien sehr klein (maximal ca. 40 bei HLT) und überschaubar ist. MYCIN hat beispielsweise 500 Regeln. Ein weiterer Grund ist, daß die Regeln kaum verschachtelt sind und die gegenseitigen Regelaufrufe höchstens eine Stufe von 5-6 erreichen. Ein dritter Grund ist schließlich, daß viele der Folgerungen von sich aus erklärt sind. Wurde z.B. die Clique {a,b,c} hergeleitet, so ist eine Rechtfertigung dafür eigentlich überflüssig, da man mit dem Ergebnis weiß, daß zwischen allen Cliquenmitgliedern M-Relationen bestehen müssen.

Ein weiteres Manko bestand darin, daß manche Prädikate und Problemstellungen algorithmischen Charakter hatten und damit die rein deklarative Betrachtung durchbrochen wurde. Es konnten nicht alle Prädikate durch Definitionsketten auf grundlegendere Prädikate zurückgeführt werden, wie dies bei den meisten typischen Expertensystemanwendungen der Fall ist. Dies machte insbesondere Probleme bei der Einordnung dieser Prädikate in das Regelschema und der Ausgabe über die Erklärungskomponente. Ein Beispiel ist der Verbesserungsalgorithmus von ART oder der Cliquensuch-Algorithmus in HLT.

Inwieweit dies ein generelles Merkmal von Theorien ist, ist aber zu bezweifeln, und wir vermuten, daß dies eher eine Eigenschaft speziell dieser Theorien ist. Es bleibt zukünftigen Projekten vorbehalten, komplexere Theorien in diesem Ansatz darzustellen mit einer umfangreicheren Regelmenge, bei denen vielleicht eine durchgängig deklarative Interpretation möglich ist.

Wir möchten anhand des letzten praktischen Beispiels noch einmal zusammenfassend auf die allgemeinen Vorteile des Computers und des regelbasierten Ansatzes für die Theorienbildung verweisen. Der Rechner bietet die Möglichkeit, Gedankenexperimente gezielt und kontrolliert durchzuführen und verschiedene Alternativen

auszuprobieren. Wir haben uns theoretisch interessante Alternativen überlegt, diese implementiert und deren theoretische Implikationen beobachtet. Das Computerprogramm hat auf bestimmte Erklärungsmöglichkeiten hingewiesen und auf bestimmte Theoriedefizite aufmerksam gemacht. Die Klärung dieser Defizite kann nicht durch Experimente am Computer erfolgen, sondern ist empirischer Forschung vorbehalten. Der Vorteil eines Computermodells liegt nicht darin, Antworten darauf zu finden, wie die Welt beschaffen ist, sondern wie sie beschaffen sein könnte. Zwar sind viele der entdeckten Zusammenhänge rein formal beweisbar, aber die computerisierte Form einer Theorie kann heuristische Anregungen geben. Dieses heuristische Experimentieren ist ein unschätzbarer Vorteil bei der Weiterentwicklung von Theorien. Im heuristischen Experimentieren in Verbindung mit der Deduktionsmächtigkeit und der Erklärung der Schlüsse liegt in unserer Sicht die Stärke maschineller, wissensbasierter Theorien und ein mächtiges Potential für die Weiterentwicklung von Theorien.

Anhang A

Verzeichnis der Symbole und Abkürzungen

Mengenlehre

Große Buchstaben werden normalerweise zur Bezeichnung von Mengen benutzt und kleine Buchstaben für Elemente von Mengen. Im Kontext von PROLOG bezeichnen wegen PROLOG-Konvention Zeichenketten, die mit Großbuchstaben beginnen sowohl Mengen als auch Elemente von Mengen.

Mengen werden extensional dargestellt durch Aufzählung aller ihrer Argumente $\{a,b,c,...\}$ oder intensional durch Angabe eines Prädikats P, das auf die Menge der Elemente zutrifft: $\{x \mid x$ hat die Eigenschaft $P\}$.

Die Elemente von Mengen sind ungeordnet, d.h. $\{a,b\} = \{b,a\}$.

Sind X und Y Mengen, dann haben die Ausdrücke der linken Seite folgende Bedeutung:

$x \in X$	x ist Element von X
$x \notin X$	x ist nicht Element von X
$\varnothing$	leere Menge $\{\}$
$\mathbb{N}$	Menge der natürlichen Zahlen
$X \cup Y$	Vereinigungsmenge von X und Y
$X \cap Y$	Durchschnittsmenge von X und Y (Zwei Mengen, deren Durchschnitt leer ist, heißen disjunkt)
$X \setminus Y$	Differenzmenge von X und Y (X ohne Y)
$X \subseteq Y$	X ist Teilmenge von Y
$X \subset Y$	X ist echte Teilmenge von Y
$\bigcup_{i=1}^{n} X_i$	Vereinigung aller n Mengen $X_1,...,X_n$
$\bigcap_{i=1}^{n} X_i$	Durchschnitt aller n Mengen $X_1,...,X_n$
$\langle x,y \rangle$	Geordnete Paare, d.h. $\langle x,y \rangle \neq \langle y,x \rangle$
$X \times Y$	Cartesisches Produkt aus X und Y (Menge, die alle geordneten Paare $\langle x,y \rangle$ mit $x \in X$ und $y \in Y$ enthält)
$\langle x_1,...,x_n \rangle$	(Geordnetes) n-Tupel
$R \subseteq X \times Y$	Relationen
Pot(X)	Potenzmenge von X (Menge aller Teilmengen)
$f: X \rightarrow Y$	Funktion f von X in Y (Relation f von X nach Y derart, daß es zu jedem $x \in X$ genau ein $y \in Y$ gibt)
card(X)	Kardinalität von X, d.h. die Anzahl der Elemente von X

Logik

Logische Ausdrücke sind in ihrer üblichen umgangssprachlichen Bedeutung zu verstehen mit den bekannten Normierungen im Fall von "wenn...dann---". Normalerweise werden die umgangssprachlichen Ausdrücke benutzt, in manchen Kontexten verwenden wir die üblichen Symbole aus der Prädikatenlogik:

... $\wedge$ ---	... und ---
... $\vee$ ---	... oder ---
$\neg$ ...	nicht ...
... $\rightarrow$ ---	wenn ... dann---
... $\leftrightarrow$ ---	... genau dann wenn --- (gdw)
$\forall$ ...	für alle ...
$\exists$ ...	es gibt (mindestens) ein ...
... = ---	... ist identisch mit ---
... $\neq$ ---	... ist nicht identisch mit ---
... := ---	... ist durch Definition identisch mit ---

Strukturalistische Symbole

M	Menge der Modelle
M_p	Menge der potentiellen Modelle
M_{pp}	Menge der partiellen potentiellen Modelle
I	Menge der intendierten Anwendungen
F(I)	Menge der festen Anwendungen
A(I)	Menge der angenommenen Anwendungen
T	Theorie(element)
K	Theoriekern
N	Theoriennetz
SC	Scientific Community (wissenschaftliche Gemeinschaft)
h	historische Periode
G	wissenschaftliche Generation
T* σ T	T* ist eine Spezialisierung von T
T ρ T*	ρ reduziert T auf T*

Anhang B

Vollständige PROLOG-Programme

Heider-Theorie (HT)

```
/*  BALANCETHEORIE VON HEIDER (HT)  */


/*  STRUKTURALISTISCHE PRAEDIKATE  */


/*  Heider-Graph (DEFINITION 1)  */

heider_graph(T,X) :-
   X= [O,P,N],
   objekte(O),
   not O=[],
   O = [O1,O2,O3],
   atom(O1),atom(O2),atom(O3),
   zeit(T_liste),
   member(T,T_liste),
   findall([X1,X2],positiv(T,X1,X2),P),
   findall([Y1,Y2],negativ(T,Y1,Y2),N),
   kreuzprodukt(O,O,Kreuz),
   teilmenge(P,Kreuz),
   teilmenge(N,Kreuz),
   durchschnitt(P,N,[]),
   append(P,N,Rel),
   irreflexiv(O,Rel),
   vollstaendig(Kreuz,Rel),!.


irreflexiv([],_).
irreflexiv([X|R],Rel) :-
   not member([X,X],Rel),
   irreflexiv(R,Rel).

vollstaendig([],_).
vollstaendig([[A,A]|Rest],Rel) :-
   vollstaendig(Rest,Rel).
vollstaendig([[A,B]|Rest],Rel) :-
   member([A,B],Rel),
   vollstaendig(Rest,Rel).
vollstaendig([[A,B]|Rest],Rel) :-
   member([B,A],Rel),
   vollstaendig(Rest,Rel).


/*  Definition einer Relation,  */
/*  Nichtperson, Person (DEFINITION 2) */


r(T,R) :-
   heider_graph(T,[_,P,N]),
   vereinigung(P,N,R).

np(T,X) :-
   heider_graph(T,[O,P,N]),
   member(X,O),
   not positiv(T,X,_),
   not negativ(T,X,_).

pe(T,X) :-
   heider_graph(T,[O,_,_]),
   np(T,Y),
   differenz(O,[Y],X_liste),
   member(X,X_liste).

relation(T,X,Y) :-
   heider_graph(T,[O,_,_]),
   member(X,O),
   member(Y,O),
   not X=Y,
   r(T,R),
   (
   member([X,Y],R)
   ;
   member([Y,X],R)
   ).


/* Triadendefinitionen (DEFINITION 3) */

triade(T,O,gleichgewicht) :-
   heider_graph(T,[O,P,N]),
   member(A,O),
   pe(T,A),
   member(B,O),
   not B = A,
   pe(T,B),
   member(C,O),
   np(T,C),
   member([A,B],P),
   member([A,C],P),
   member([B,C],P),!.

triade(T,O,gleichgewicht) :-
   heider_graph(T,[O,P,N]),
   member(A,O),
   pe(T,A),
   member(B,O),
   not B = A,
   pe(T,B),
   member(C,O),
   np(T,C),
   member([A,B],P),
   member([A,C],N),
   member([B,C],N),!.

triade(T,O,gleichgewicht) :-
   heider_graph(T,[O,P,N]),
   member(A,O),
   pe(T,A),
   member(B,O),
   not B = A,
   pe(T,B),
   member(C,O),
   np(T,C),
   member([A,B],N),
   member([A,C],N),
   member([B,C],P),!.

triade(T,O,gleichgewicht) :-
   heider_graph(T,[O,P,N]),
   member(A,O),
   pe(T,A),
   member(B,O),
   not B = A,
   pe(T,B),
   member(C,O),
   np(T,C),
   member([A,B],N),
   member([A,C],P),
   member([B,C],N),!.
```

```
triade(T,O,ungleichgewicht) :-
   heider_graph(T,[O,P,N]),
   member(A,O),
   pe(T,A),
   member(B,O),
   not B = A,
   pe(T,B),
   member(C,O),
   np(T,C),
   member([A,B],P),
   member([A,C],P),
   member([B,C],N),!.

triade(T,O,ungleichgewicht) :-
   heider_graph(T,[O,P,N]),
   member(A,O),
   pe(T,A),
   member(B,O),
   not B = A,
   pe(T,B),
   member(C,O),
   np(T,C),
   member([A,B],P),
   member([A,C],N),
   member([B,C],P),!.

triade(T,O,ungleichgewicht) :-
   heider_graph(T,[O,P,N]),
   member(A,O),
   pe(T,A),
   member(B,O),
   not B = A,
   pe(T,B),
   member(C,O),
   np(T,C),
   member([A,B],N),
   member([A,C],P),
   member([B,C],P),!.

triade(T,O,unbestimmt) :-
   heider_graph(T,[O,P,N]),
   member(A,O),
   pe(T,A),
   member(B,O),
   not B = A,
   pe(T,B),
   member(C,O),
   np(T,C),
   member([A,B],N),
   member([A,C],N),
   member([B,C],N),!.


/* Potent.Modell von HT (DEFINIT. 4) */

pot_modell(X) :-
   X=[O,T,'<',Pt,Nt],
   objekte(O), write(O),
   O=[O1,O2,O3],
   atom(O1),atom(O2),atom(O3),
   zeit(T_liste),
   lineare_ordnung(T_liste,'<'),
   kreuzprodukt(O,O,Kreuz),
   potenzmenge(Kreuz,Potmenge),
   funktion(positiv,T_liste,Pt,Potmenge),
   funktion(negativ,T_liste,Nt,Potmenge),
   !,
   sind_alle_heider_graphen(T_liste,
   [O,Pt,Nt]).

lineare_ordnung([X,Y], Rel) :-
   Z =.. [Rel,X,Y],
   call(Z).
lineare_ordnung([X,Y|R],Rel) :-
   Z =..[Rel,X,Y],
   call(Z),
   lineare_ordnung([Y|R],Rel).

funktion(Rel,[],[],_).
funktion(Rel,[T|T_rest],[(T,L)|R1],
   Potmenge)
   :-
   Z =.. [Rel,T,X1,X2],
   findall([X1,X2],Z,L),
   teilmenge(L,Potmenge),
   funktion(Rel,T_rest,R1,Potmenge).

sind_alle_heider_graphen([],[_,[],[]]).
sind_alle_heider_graphen([T|T_rest],O,
   [(T,P)|P_rest],[(T,N)|N_rest]])
   :-
   heider_graph(T,[O,P,N]),!,
   sind_alle_heider_graphen(T_rest,
   [O,P_rest,N_rest]).



/*  Modell von HT  (DEFINITION 6)  */

modell(X) :-
   X=[O,T_liste,_,_,_],
   pot_modell(X),!,
   axiom_3_test(T_liste, T_liste,O).

axiom_3_test([],_,_).
axiom_3_test([T1|T_rest],T_liste,O) :-
   triade(T1,O,ungleichgewicht),
   member(T2,T_liste),
   T1 < T2,
   triade(T2,O,gleichgewicht),!,
   axiom_3_test(T_rest,T_liste,O).
axiom_3_test([T1|T_rest],T_liste,O) :-
   triade(T1,O,gleichgewicht),
   member(T2,T_liste),
   T1 >= T2,
   triade(T2,O,ungleichgewicht),!,
   axiom_3_test(T_rest,T_liste,O).



/*  MENGENSPRACHLICHE PRAEDIKATE  */


/*  Argument ist eine Liste  */

liste([]).
liste([_|_]).


/*  Differenzmenge  */

differenz(L,[],L) :- !.
differenz([Kopf|Rest],L,U) :-
   member(Kopf,L),!,
   differenz(Rest,L,U).
differenz([Kopf|Rest],L,[Kopf|Rest2]) :-
   !,
   differenz(Rest,L,Rest2).
differenz(_,_,[]).


/*  Teilmenge  */

teilmenge([],Y).
teilmenge([K|X],Y) :-
   member(K,Y),
   teilmenge(X,Y).


/*  Schnittmenge  */

durchschnitt([],X,[]).
durchschnitt([K|R],Y,[K|Z]) :-
   member(K,Y),!,
   durchschnitt(R,Y,Z).
```

```
durchschnitt([K|R],Y,Z) :-
   durchschnitt(R,Y,Z).


/*  Vereinigungsmenge  */

vereinigung([],X,X).
vereinigung([K|R],Y,Z) :-
   member(K,Y),!,
   vereinigung(R,Y,Z).
vereinigung([K|R],Y,[K|Z]) :-
   vereinigung(R,Y,Z).


/*  Cartesisches Produkt  */

kreuzprodukt(X,Y,Z) :-
   listall(K,pair(X,Y,K),Z1),
   mkset(Z1,Z), !.

kreuzprodukt(X,Y,Z) :-
   send_msg(kreuzprodukt,sup,fail,
   kreuzprodukt(X,Y,Z)).

send_msg(FromWhom,sup,fail,Msg) :-
   assertz(msg(FromWhom,sup,fail,Msg)).
send_msg(FromWhom,sup,failfail,Msg) :-
   assertz(msg(FromWhom,sup,fail,Msg)),
   !,fail .
send_msg(FromWhom,sup,Type,Msg) :-
   assertz(msg(FromWhom,sup,Type,Msg)).

listall(X,G,_) :-
   asserta(found(mark)),
   call(G),
   asserta(found(X)),
   fail.
listall(_,_,L) :-
   collect([],M),
   !, L = M.

collect(S,L) :-
   collget(X),
   !, collect([X|S],L).
collect(L,L).

collget(X) :-
   retract(found(X)),
   !, X \= mark.

pair(X,Y,K) :- member(A,X),
   member(B,Y),
   K = [A,B].

mkset([],[]).
mkset([X|L],Z) :-
member(X,L),!,mkset(L,Z).
mkset([X|L],[X|Z]) :- mkset(L,Z).


/*  Potenzmenge  */

potenzmenge(X,Y) :-
   potset(X,Y).

potset([],[]).
potset(M,N) :-
   card(M,L),
   potset(M,L,N).

potset(M,0,[]) :- ! .
potset(M,I,N) :-
   genvarlst(I,N),
   potset1(M,N).
potset(M,I,N) :-
   I1 is I-1,
   potset(M,I1,N).

potset1(_,[]) :- ! .
potset1([F|R],[M1|V]) :-
   potmember(M1,[F|R],U),
   potset1(U,V).

potmember(X,[X|Y],Y).
potmember(X,[_|Y],Z) :-
   potmember(X,Y,Z).

genvarlst(0,[]) :- ! .
genvarlst(I,[F|R]) :-
   I1 is I-1,
   genvarlst(I1,R).


card([],0).
card([K|Rest],N) :-
   card(Rest,N1),
   N is N1 + 1.



/*  SCHNITTSTELLE ZUM BENUTZER  */

hilfe :-
   nl,nl,nl,
   write('* ********** HEIDER-THEORIE -
          STRUKTURALISTISCH INTERPRETIERT
          ********** *'),nl,nl,
   write('Es wird das Vorliegen folgender
          Daten erwartet:'),nl,
   write('Menge der Objekte.
          Einlesen als Liste mit
          -> lese_objekte'),nl,
   write('Menge der Zeitpunkte.
          Einlesen als Liste mit
          -> lese_zeit'),nl,
   write('Menge der Relationen.
          Einlesen als Subj-Praed-Obj-
          Saetze
          -> lese_relationen'),nl,
   write('Neue Faktenbasis
          -> neu.'),nl,
   write('Ueberblick vorhandene Fakten
          -> fakten'),nl,nl.


fakten :-
   listing(objekte),
   listing(zeit),
   listing(positiv),
   listing(negativ).


/*  EINLESEN DER DATEN  */

neu :-
   abolish(zeit(_)),
   abolish(objekte(_)),
   abolish(positiv(_,_,_)),
   abolish(negativ(_,_,_)),
   asserta(positiv(-1,dummy,dummy)),
   asserta(negativ(-1,dummy,dummy)).

positiv(-1,dummy,dummy).
negativ(-1,dummy,dummy).

lese_zeit :-
   write('Zeitpunkte in einer Liste
   angeben'),nl,
   write('> '),
   read(L),
   liste(L),
   pruefe_liste(L,integer),
   asserta(zeit(L)),!.
lese_zeit :-
   write('Eingabefehler').
```

```
lese_objekte :-
   write('Objekte in einer Liste
   angeben'),nl,
   write('> '),
   read(L),
   pruefe_liste(L,atom),
   asserta(objekte(L)),!.
lese_objekte :-
   write('Eingabefehler').

lese_relationen :-
   write('Zeitpunkt (e=Ende) > '),
   read(Z),
   not member(Z,[ende,'Ende',end,e]),
   write('Fakt (Subj-Praed-Obj) >'),
   read_in(L),
   bereite_auf(Z,L,Rel),
   asserta(Rel),
   lese_relationen.

lese_relationen :-
   write('Fertig').

bereite_auf(Z,L,Rel) :-
   integer(Z),
   entferne_muell(L,[S,Pr,O]),
   klassifiziere(Pr,Klasse),!,
   Rel =.. [Klasse,Z,S,O].
bereite_auf(_,_,_) :-
   write('Fehler in Eingabe.').

entferne_muell([],[]).
entferne_muell([A|Rest],Erg) :-
   trash(Liste),
   member(A,Liste),
   entferne_muell(Rest,Erg).
entferne_muell([A|Rest],[A|Erg]) :-
   entferne_muell(Rest,Erg).

klassifiziere(Relation,Typ) :-
   klasse(Typ,Relliste),
   member(Relation,Relliste),!.

klassifiziere(Relation,Typ) :-
   frage(relation(Relation)),
   read(Antw),
   erkenne(Antw,Typ),
   klassifiziere(Relation,Typ).

frage(relation(Relation)) :-
   schreib_liste(
   ['Relation',Relation,' ist nicht im
     Lexikon','Ist die Relation positiv,
     negativ oder indifferent? ']).

erkenne(Antw,Typ) :-
   bedeutet(Antw,positiv).
erkenne(Antw,Typ) :-
   bedeutet(Antw,negativ).

erkenne(Ant,Typ) :-
    write('Ich kann die Eingabe nicht
          verstehen.'),nl,
    write('Bitte wiederholen Sie die
          Antwort.'),
    erkenne(Antw,Typ).

bedeutet(p,positiv).
bedeutet(pos,positiv).
bedeutet(positiv,positiv).
bedeutet(n,negativ).
bedeutet(neg,negativ).
bedeutet(negativ,negativ).

klasse(positiv,[plus,p,positiv,pos,likes,
                like]).
klasse(negativ,[negativ,n,m,minus,min,
                neg,hates]).

trash(['.',',',';',':','?','!']).


/* Einlesen in Liste (Clocksin/Mellish
   1987) */


read_in([W|Ws]) :-
   get0(C),
   readword(C,W,C1),
   restsent(W,C1,Ws).

restsent(W,_,[]) :-
   lastword(W),!.
restsent(W,C,[W1|Ws]) :-
   readword(C,W1,C1),
   restsent(W1,C1,Ws).

readword(C,W,C1) :-
   single_characters(C),!,
   name(W,[C]),
   get0(C1).
readword(C,W,C2) :-
   in_word(C,NewC),!,
   get0(C1),
   restword(C1,Cs,C2),
   name(W,[NewC|Cs]).
readword(C,W,C2) :-
   get0(C1),
   readword(C1,W,C2).

in_word(C,C) :- C>96, C<123.
in_word(C,L) :- C>64, C<91, L is C+32.
in_word(C,C) :- C>47, C<58.
in_word(39,39).
in_word(45,45).

restword(C,[NewC|Cs],C2) :-
   in_word(C,NewC),!,
   get0(C1),
   restword(C1,Cs,C2).
restword(C,[],C).

single_characters(44).
single_characters(59).
single_characters(58).
single_characters(63).
single_characters(33).
single_characters(46).

lastword('.').
lastword('!').
lastword('?').


/*  ALLGEMEINE HILFSPRAEDIKATE  */

pruefe_liste([],_).
pruefe_liste([X|Rest],Rel) :-
   Z =.. [Rel,X],
   call(Z),
   pruefe_liste(Rest,Rel).


schreib_liste([]) :- nl.
schreib_liste([Kopf|Rest]) :-
   write(Kopf),
   tab(1),
   schreib_liste(Rest).
```

Abelson-Rosenberg-Theorie (ART)

```
/*  BALANCETHEORIE VON ABELSON/ROSENBERG
    (ART) */


/*  INFERENZMASCHINE  */


leite_ab(Konkl) :-
   faktum(_,Konkl).

leite_ab(Konkl) :-
   regel(Nr,Konkl,Beding),
   not faktum(_,Konkl),
   assertz(faktum([Nr|Beding],Konkl)),
   schreib_liste([Konkl,
   ' abgeleitet mit Regelnr ',Nr]).

leite_ab :-
   regel(Nr,Konkl,Beding),
   not faktum(_,Konkl),
   assertz(faktum([Nr|Beding],Konkl)),
   schreib_liste([Konkl,
   ' abgeleitet mit Regelnr ',Nr]),
   fail.


/*  PSYCHO-LOGIK-REGELN  */


regel(1,p(A,C),[p(A,B),p(B,C)]) :-
   faktum(empir,p(A,B)),
   faktum(empir,p(B,C)).

regel(2,n(A,C),[p(A,B),n(B,C)]) :-
   faktum(empir,p(A,B)),
   faktum(empir,n(B,C)).

regel(3,p(A,C),[n(A,B),n(B,C)]) :-
   faktum(empir,n(A,B)),
   faktum(empir,n(B,C)).

regel(4,a(A,C),[p(A,C),n(A,C)]) :-
   faktum([_|_],p(A,C)),
   faktum([_|_],n(A,C)).

regel(5,a(A,C),[p(A,C),n(A,C)]) :-
   faktum(empir,p(A,C)),
   faktum([_|_],n(A,C)).

regel(6,a(A,C),[p(A,C),n(A,C)]) :-
   faktum([_|_],p(A,C)),
   faktum(empir,n(A,C)).

regel(7,ungleichgewicht,[a(A,B)]) :-
   faktum([_|_],a(A,B)).

regel(8,gleichgewicht,['nicht es gibt
   x,y: a(x,y)'])
   :-
   not faktum([_|_],a(_,_)).


/*  ERKLAERUNG  */


wie :-
   findall(Konkl,faktum([Nr|_],Konkl),
   K_Liste),
   konkl_liste(K_Liste).

konkl_liste([]).
konkl_liste([Kopf|Rest]) :-
   wie(Kopf),
   konkl_liste(Rest).


wie(Konkl) :-
   faktum([Nr|Bed_liste],Konkl),
   write('\n\nEmpirische Relationen: ' ),
   schreib_liste(Bed_liste),
   gib_regel_aus(Nr),
   write('Also: '),
   write(Konkl).


gib_regel_aus(Nr) :-
   asserta(schon_vergeben(dummy)),
   clause(regel(Nr,Konkl,_), Ursachen),
   schreib_liste(['Regel',Nr,')']),
   write('WENN '),
   schreibe_ursachen(Ursachen),
   write(' DANN '),
   schreibe_konklusion(Konkl),
   abolish(schon_vergeben(_)).


schreibe_konklusion(Konkl) :-
   instantiiere(Konkl),
   write(Konkl),nl,nl.


schreibe_ursachen(filter(_,_,_)).
schreibe_ursachen(faktum(Typ,Fakt)) :-
   instantiiere(Fakt),
   write(Fakt),
   pruefe_ob_regel(Typ,_,X),
   tab(1),write(X).
schreibe_ursachen(faktum(Typ,Fakt)','
   Ursachen)
   :-
   instantiiere(Fakt),
   write(Fakt),
   pruefe_ob_regel(Typ,_,X),
   tab(1), write(X),
   write(' UND '),
   schreibe_ursachen(Ursachen).
schreibe_ursachen(not faktum(Typ,Fakt))
   :-
   write(' NOT '),
   schreibe_ursachen(faktum(Typ,Fakt)).

pruefe_ob_regel([Rnr|Rest],Rnr,
   '(ABGELEITET)')
   :- !.
pruefe_ob_regel(Emp,Emp,' ').


instantiiere(X) :-
   X=..L,
   pruefe_instanz(L,L1),
   X=..L1.

pruefe_instanz([],[]).
pruefe_instanz([X|R],[X|R2]) :-
   var(X),
   variablenliste(L),
   member(V,L),
   not schon_vergeben(V),
   X = V,
   asserta(schon_vergeben(V)), !,
   pruefe_instanz(R,R2).
```

```
pruefe_instanz([X|R],[X|R2]) :-
   nonvar(X),
   pruefe_instanz(R,R2).


variablenliste([x,y,z]).

regeln :-
   alle_regeln(1).

alle_regeln(9) :-
   nl,write('OK - All rules').
alle_regeln(N) :-
   gib_regel_aus(N),
   N1 is N+1,
   alle_regeln(N1).


zeige_widersprueche :-
   faktum(L1,Fakt1),
   faktum(L2,Fakt2),
   Fakt1 =.. [R1,A,B],
   Fakt2 =.. [R2,A,B],
   not (R1 = R2),
   not (R1 = a),
   not (R2 = a),
   pruefe_ob_regel(L1,Erg1,_),
   pruefe_ob_regel(L2,Erg2,_),
   write(Fakt1),
   write(' (') ,write(Erg1),write(') '),
   write('  inkonsistent mit  '),
   write(Fakt2),
   write(' (') ,write(Erg2),
   write(') '),nl,
   fail.



/*  MATRIXOPERATIONEN  */


/* Matrix generieren */

generiere_matrix :-
   objektpaare(X),
   erzeuge_matrix(X,0).

erzeuge_matrix([],_).
erzeuge_matrix([[A,B],[A,C]|Rest],N) :-
   faktum(empir,X),
   X=..[R,A,B],
   assertz(found(R)),
   erzeuge_matrix([[A,C]|Rest],N).
erzeuge_matrix([Kopf|Rest],N) :-
   faktum(empir,X),
   X=..[R|Kopf],
   assertz(found(R)),
   findall(Rel,found(Rel),Liste),
   N1 is N + 1,
   assertz(matrix(N1,Liste)),
   assertz(matrix_alt(N1,Liste)),
   abolish(found(_)),
   erzeuge_matrix(Rest,N1).


/* Matrix ausgeben */


matrix :-
   matrix_alt(N,Matr),
   write(N),tab(1),schreib_liste(Matr),
   fail.
matrix.
```

```
/* Balancegenerierungs-Algorithmus */

/* zaehlt n oder p in Vektor */
zaehle_np(_,[],0).
zaehle_np(Np,[Np|Rest],Zahl) :-
   zaehle_np(n,[p,n,n],Erg)
   zaehle_np(Np,Rest,N1),
   Zahl is 1 + N1.
zaehle_np(Np,[_|Rest],Zahl) :-
   zaehle_np(Np,Rest,Zahl).


/* berechnet Differenz n/p in Vektor */
zaehle_vektor([],[]).
zaehle_vektor([Kopf|Rest],[Diff|Ergeb])
   :-
   zaehle_np(n,Kopf,N_ergeb),
   zaehle_np(p,Kopf,P1_ergeb),
   P_ergeb is P1_ergeb - 1,
   Diff is N_ergeb - P_ergeb,
   zaehle_vektor(Rest,Ergeb).


/* Hauptpraedikat Matrixmanipulation */
verbessere_matrix :-
   findall(X,matrix(_,X),Liste),
   write('Alte Matrix S: '),nl,
   schreib_liste_nl(Liste),
   zaehle_vektor(Liste,Diff_liste),
   maxlist(Diff_liste,Max),
   indx(Diff_liste,Max,Indx),!,
   Max >= 0,
   grenzwert_oben(Oben),
   Max =< Oben,
   retract(grenzwert_oben(Oben)),
   asserta(grenzwert_oben(Max)),
   wechsle_np(Indx,0),
   findall(X,matrix(_,X),L2),
   schreib_liste(['Differenz N - P: ',
   Max, ' in Zeile/Spalte: ',Indx]),
   write('Neue Matrix S*: '),nl,
   schreib_liste_nl(L2),!.

verbessere_matrix :-
   abolish(grenzwert_oben(_)),
   asserta(grenzwert_oben(10000)),
   write('Keine Verbesserung mehr
   moeglich').


maxlist([Max],Max).
maxlist([K1,K2|Rest],Max) :-
   maxlist([K2|Rest],Max1),
   max(K1,Max1,Max).


max(X,Y,X) :- X >= Y.
max(X,Y,Y) :- X < Y.


indx([K],K,1) :- !.
indx([K|Rest],K,1) :- !.
indx([_|Rest],K,N) :-
   indx(Rest,K,N1),
   N is N1 + 1.

wechsle_np(_,X) :-
   objektzahl(Y),
   X is Y + 1.
wechsle_np(Indx,Laufvar) :-
   N is Laufvar + 1,
   Indx = N,
   matrix(N,Matr),
   wechsle_all(Indx,0,Matr,Erg),
   retract(matrix(N,Matr)),
   assertz(matrix(N,Erg)),
   wechsle_np(Indx,N).
```

```
wechsle_np(Indx,Laufvar) :-
   N is Laufvar + 1,
   matrix(N,Matr),
   wechsle1_np(Indx,0,Matr,Erg),
   retract(matrix(N,Matr)),
   assertz(matrix(N,Erg)),
   wechsle_np(Indx,N).


wechsle_all(Indx,N,[],[]).
wechsle_all(Indx,N,[K|Rest],[K|Erg]) :-
   N1 is N + 1,
   Indx = N1,
   wechsle_all(Indx,N1,Rest,Erg).
wechsle_all(Indx,N,[p|Rest],[n|Erg]) :-
   N1 is N + 1,
   wechsle_all(Indx,N1,Rest,Erg).
wechsle_all(Indx,N,[n|Rest],[p|Erg]) :-
   N1 is N + 1,
   wechsle_all(Indx,N1,Rest,Erg).
wechsle_all(Indx,N,[o|Rest],[o|Erg]) :-
   N1 is N + 1,
   wechsle_all(Indx,N1,Rest,Erg).

wechsle1_np(Indx,N,[],[]).
wechsle1_np(Indx,N,[Pn|Rest],[Np|Erg]) :-
   N1 is N + 1,
   Indx = N1,
   change(Pn,Np),
   wechsle1_np(Indx,N1,Rest,Erg).
wechsle1_np(Indx,N,[K|Rest],[K|Erg]) :-
   N1 is N + 1,
   wechsle1_np(Indx,N1, Rest,Erg).

change(p,n).
change(n,p).
change(o,o).



/*  DATENERHEBUNG  */


/*  Hauptpraedikate  */

lesen :-
   neu,
   lo,
   gk,
   lr.

lo :-
   lies_objekte.
lr :-
   lies_relationen.
gk :-
   generiere_kreuz.


lies_objekte :-
   write('Eingabe der Objekte > '),
   read_in(Rohobjekte),
   bereinige(Rohobjekte,Objekte),
   cons(ego,Objekte,Objektliste),
   laenge(Objekte,Zahl),
   asserta(objektzahl(Zahl)),
   asserta(objekte(Objektliste)).

generiere_kreuz :-
   objekte(Objektliste),
   cross(Objektliste,Objektliste,Kreuz),
   asserta(objektpaare(Kreuz)).

lies_relationen :-
   objektpaare(Paarliste),
   write('Eingabe der Relationen '),nl,
   lies_rel(Paarliste).


lies_rel([]).
lies_rel([[A,A]|Rest]) :-
   assertz(faktum(empir,p(A,A))),
   lies_rel(Rest).
lies_rel([[A,B]|Rest]) :-
   symmetrie(Liste),
   member_1([X,A,B],Liste),
   F=..[X,A,B],
   assertz(faktum(empir,F)),
   lies_rel(Rest).
lies_rel([[A,B]|Rest]) :-
   schreib_liste(['Relation ',A,B]),
   write('> '),
   read(Relation),
   klassifiziere(Relation,X),
   F=..[X,A,B],
   write(F),nl,
   assertz(faktum(empir,F)),
   trage_ein_sym([X,B,A]),!,nl,
   lies_rel(Rest).


/*  Hilfspraedikate zur Datenerhebung  */

bereinige([],[]).
bereinige([A|Rest],Erg) :-
   trash(Liste),
   member_1(A,Liste),
   bereinige(Rest,Erg).
bereinige([A|Rest],[A|Erg]) :-
   bereinige(Rest,Erg).

trage_ein_sym(Sym) :-
   symmetrie(Liste),
   cons(Sym,Liste,Erg),
   retract(symmetrie(Liste)),
   assertz(symmetrie(Erg)).



/*  Klassifikation  */

klassifiziere(Relation,Typ) :-
   klasse(Typ,Relliste),
   member(Relation,Relliste).

klassifiziere(Relation,Typ) :-
   frage(relation(Relation)),
   read(Antw),
   erkenne(Antw,Typ),
   retract(klasse(Typ,Liste)),
   asserta(klasse(Typ,[Relation|Liste])),
   klassifiziere(Relation,Typ).

klasse(p,[p,plus,positiv,pos,likes,like,
          supports]).
klasse(n,[n,negativ,minus,min,neg,
          dislikes,hates]).
klasse(o,[o,neutral,weiss_nicht,
          ambivalent]).

frage(relation(Relation)) :-
   schreib_liste_nl(['Relation',Relation,
   ' ist nicht im Lexikon',
   'Ist die Relation positiv, negativ
    oder indifferent? ']).


erkenne(Antw,Typ) :-
   bedeutet(Antw,p).
erkenne(Antw,Typ) :-
   bedeutet(Antw,n).
erkenne(Antw,Typ) :-
   bedeutet(Antw,o).
```

```
erkenne(Antw,Typ) :-
   write('Ich kann die Eingabe nicht
   verstehen.'),nl,
   write('Bitte wiederholen Sie die
   Antwort.'),
   erkenne(Antw,Typ).

bedeutet(p,p).
bedeutet(pos,p).
bedeutet(positiv,p).
bedeutet(n,n).
bedeutet(neg,n).
bedeutet(negativ,n).
bedeutet(i,o).
bedeutet(ind,o).
bedeutet(indifferent,o).


/*  Einlesen in Liste (Clocksin/Mellish
1987) */


read_in([W|Ws]) :-
   get0(C),
   readword(C,W,C1),
   restsent(W,C1,Ws).

restsent(W,_,[]) :-
   lastword(W),!.
restsent(W,C,[W1|Ws]) :-
   readword(C,W1,C1),
   restsent(W1,C1,Ws).

readword(C,W,C1) :-
   single_characters(C),!,
   name(W,[C]),
   get0(C1).
readword(C,W,C2) :-
   in_word(C,NewC),!,
   get0(C1),
   restword(C1,Cs,C2),
   name(W,[NewC|Cs]).
readword(C,W,C2) :-
   get0(C1),
   readword(C1,W,C2).

in_word(C,C) :- C>96, C<123.
in_word(C,L) :- C>64, C<91, L is C+32.
in_word(C,C) :- C>47, C<58.
in_word(39,39).
in_word(45,45).

restword(C,[NewC|Cs],C2) :-
   in_word(C,NewC),!,
   get0(C1),
   restword(C1,Cs,C2).
restword(C,[],C).

single_characters(44).
single_characters(59).
single_characters(58).
single_characters(63).
single_characters(33).
single_characters(46).

lastword('.').
lastword('!').
lastword('?').


/*  MENGENTHEORETISCHE HILFSPRAEDIKATE */


/* Member-Relation (einmal erfuellbar */

member_1(Element,[Element|_]) :- !.
member_1(Element,[_|Rest]) :-
   member_1(Element,Rest).

/*  Cartesisches Produkt  */

cross(X,Y,Z) :-
   listall(K,pair(X,Y,K),Z1),
   mkset(Z1,Z).

cross(X,Y,Z) :-
   send_msg(cross,sup,fail,cross(X,Y,Z)).


send_msg(FromWhom,sup,fail,Msg) :-
   assertz(msg(FromWhom,sup,fail,Msg)).
send_msg(FromWhom,sup,failfail,Msg) :-
   assertz(msg(FromWhom,sup,fail,Msg)),
   !,fail .
send_msg(FromWhom,sup,Type,Msg) :-
   assertz(msg(FromWhom,sup,Type,Msg)).

listall(X,G,_) :-
   asserta(found(mark)),
   call(G),
   asserta(found(X)),
   fail.
listall(_,_,L) :-
   collect([],M),
   !, L = M.

collect(S,L) :-
   collget(X),
   !, collect([X|S],L).
collect(L,L).

collget(X) :-
   retract(found(X)),
   !, X \= mark.

pair(X,Y,K) :-
   member(A,X),
   member(B,Y),
   K = [A,B].

mkset([],[]).
mkset([X|L],Z) :-
   member(X,L),!,
   mkset(L,Z).
mkset([X|L],[X|Z]) :-
   mkset(L,Z).


/* Element einfügen */

cons(A1,[],[A1]).
cons(A1,[H|T],[A1,H|T]).


/* Laenge einer Liste */

laenge([],0).
laenge([_|Rest],N) :-
   laenge(Rest,N1),
   N is 1 + N1.
```

```
/*  ALLGEMEINE HILFSPRAEDIKATE  */

hilfe :-
   nl,nl,
   write(' ********* BALANCETHEORIE VON
          ABELSON/ROSENBERG (ART, 1958)
          ******** *'),nl,nl,
   write('  Moegliche Eingaben:'),nl,nl,
   write('  DATENBASIS AENDERN'),nl,
   write('- Daten eingeben
            -> lesen.'),nl,
   write('- Alte Wissensbasis voellig
            loeschen
            ->          neu.'),nl,
   write('- Nur Schlussfolgerungen
            loeschen
            -> entferne(schluesse).'),nl,
   write('- Matrix loeschen
            -> entferne(matrix).'),nl,
   write('- Einzelnen Fakt entfernen
            -> entferne(fakt).'),nl,
   write('- Einzelnen Fakt hinzunehmen
            -> fuege_hinzu(fakt).'),nl,
   write('  FAKTEN UND SCHLUESSE'),nl,
   write('- Alle Schluesse generieren
            -> leite_ab.'),nl,
   write('- Einzelne Fakten beweisen
            -> leite_ab(fakt).'),nl,
   write('- Ueberblick ueber gueltige
            Fakten
            -> fakten.'),nl,
   write('- Ueberblick ueber vorhandene
            Objekte
            -> objekte.'),nl,
   write('- Widerspruechliche Fakten
            -> zeige_wiedersrueche.'),nl,
   write('- Schlussrechtfertigung fuer
            Fakt:
            -> wie(fakt).'),nl,
   write('- Schlussrechtfertigung fuer
            alle Fakten
            -> wie.'),nl,
   write('- Regeln ausgeben
            -> regeln.'),nl,
   write('  MATRIZENOPERATIONEN/
            BALANCEGENERIERUNG'),nl,
   write('- Strukturmatrix erzeugen
            -> generiere_matrix.'),nl,
   write('- Aktuelle Matrix ausgeben
            -> matrix.'),nl,
   write('- Folge verbesserter Matrizen
            erzeugen
            -> verbessere_matrix.'),nl.



neu :-
   loesche,
   init.

loesche :-
   abolish(objekte(_)),
   abolish(objektzahl(_)),
   abolish(objektpaare(_)),
   abolish(faktum(_,_)),
   abolish(matrix_alt(_,_)),
   abolish(matrix(_,_)),
   abolish(grenzwert_oben(_)),
   write('Alle Datenpraedikate
   geloescht'),nl.

init :-
   asserta(faktum(dummy,dummy)),
   asserta(grenzwert_oben(10000)),
   write('Initialisierungs-Praedikate
   OK'),nl.

entferne(schluesse) :-
   retract(faktum([_|_],_)),
   fail.
entferne(schluesse) :- !.

entferne(matrix) :-
   abolish(matrix_alt(_,_)),
   abolish(matrix(_,_)),!.

entferne(Fakt) :-
   bestandteile_vorhanden(Fakt),
   faktum(_,Fakt),!,
   retract(faktum(_,Fakt)),
   entferne(schluesse),
   write('Alle Schluesse entfernt').
entferne(_) :-
   write('Falsche Eingabe oder
   Fakt nicht vorhanden').

fuege_hinzu(Fakt) :-
   bestandteile_vorhanden(Fakt),
   not faktum(_,Fakt),!,
   asserta(faktum(empir,Fakt)),
   entferne(schluesse),
   write('Alle Schluesse entfernt').
fuege_hinzu(Fakt) :-
   write('Falsche Eingabe oder Fakt
   schon da').

bestandteile_vorhanden(Fakt) :-
   Fakt =..[R,A,B],
   objekte(L),
   member(A,L),
   member(B,L),
   member(R,[p,n,o]),!.

fakten :-
   (
   write('\nEmpirisch: \n'),
   faktum(empir,Fakt),
   write(Fakt),nl,
   fail
   ;
   write('\nAbgeleitet: \n'),
   faktum([Nr|_],Fakt),
   write(Fakt),tab(3),write(Nr),nl,
   fail
   ;
   true
   ).

objekte :-
   objekte(X),
   schreib_liste(X).

trash(['.',',',';',':','?','!']).

symmetrie([]).


schreib_liste([]) :- nl.
schreib_liste([Kopf|Rest]) :-
   write(Kopf),
   tab(1),
   schreib_liste(Rest).

schreib_liste_nl([]) :- nl.
schreib_liste_nl([Kopf|Rest]) :-
   write(Kopf),nl,
   schreib_liste_nl(Rest).
```

Holland-Leinhardt-Theorie (HLT, T-Graph-Modell)

```
:-
write('\n\n**********************\n\n'),
write('HLT / T-Graph-Modell \n\n'),
write('**************************\n').


/*  DATENERHEBUNG  */


/* Daten einlesen - Tastat. od. Datei? */

lesen :-
   nl,
   write('Daten einlesen von'),nl,
   write('(T)astatur'),nl,
   write('(D)atei'),nl,nl,
   write('> '),
   zeichen_lesen(C),
   (
   C == d,
   datei,!
   ;
   C == t,
   tastatur,!
   ;
   write('Eingabefehler')
   ).


zeichen_lesen(C) :-
   get0(Cn),
   name(C,[Cn]),
   skip(10),!.


/* Daten einlesen von Tastatur */

tastatur :-
   write('Knoten eingeben'),nl,
   read_in(Rohobjekte),
   bereinige(Rohobjekte, Objekte),
   assertz(knoten(Objekte)),
   write('Objekte -> '),
   write(Objekte),nl,
   cross(Objekte,Objekte,Cross),
   assertz(kreuzprodukt(Cross)),
   write('Kreuzprodukt -> '),
   write(Cross),nl,
   read_rel(Cross).

read_rel([]).
read_rel([[A,A]|Rest]) :-
   read_rel(Rest).
read_rel([[A,B]|Rest]) :-
   nl,
   write(A),write(' -> '), write(B),
   write('\nRelation R gegeben? '),
   read(Antwort),
   Antwort = j,
   assertz(faktum(r(A,B),empirisch)),
   read_rel(Rest).
read_rel([_|Rest]) :-
   read_rel(Rest).



/* Daten einlesen von Datei */

datei :-
   asserta(knoten([])),
   lies_datei_name(Datei),
   lesen_von(Datei).

lies_datei_name(Datei) :-
   write('Dateiname: '),
   read(Datei).

lesen_von(Datei) :-
   see(Datei),
   bearbeite_datei,
   seen.

bearbeite_datei :-
   read(Term),
   bearbeite(Term).

bearbeite(end_of_file) :-
   nl,
   write('Daten eingelesen'),
   nl,!.

bearbeite(r(X,Y)) :-
   nl,write(r(X,Y)),
   asserta(faktum(r(X,Y),empirisch)),
   bearbeite_datei.

bearbeite(Liste) :-
   is_list(Liste),
   knoten(O_liste),
   retract(knoten(O_Liste)),
   write(Liste),
   asserta(knoten(Liste)),
   bearbeite_datei.

bearbeite(Element) :-
   atom(Element),
   knoten(Liste),
   write(Element),
   tab(1),
   retract(knoten(Liste)),
   asserta(knoten([Element|Liste])),
   bearbeite_datei.


/* Datenbasis entleeren */

neu :-
   abolish(faktum,2),
   abolish(knoten,1),
   write('Faktenbasis entfernt. OK.').


/* Daten entfernen, hinzufuegen */

entferne(alle_schluesse) :-
   retract(faktum(X,[_|_])),
   schreibe_liste(['Entferne ',X]),
   fail.
entferne(alle_schluesse) :-
   write('Bestehende Schlussfolgerungen
   entfernt. OK.\n').

entferne(korr_daten) :-
   entferne(alle_schluesse),
   fail.
entferne(korr_daten) :-
   retract(faktum(r(X,Y),korrigiert)),
   schreibe_liste(['Entferne ',r(X,Y)]),
   fail.
entferne(korr_daten) :-
   write('Korrigierte Daten entfernt.
   OK.\n').
```

```
entferne(r(X,Y)) :-
   faktum(r(X,Y),empirisch),!,
   retract(faktum(r(X,Y),empirisch)),
   schreibe_liste([r(X,Y),' aus der
   Datenbasis entfernt']),
   entferne(alle_schluesse).

entferne(r(X,Y)) :-
   schreibe_liste([r(X,Y), ' ist nicht
   in der Datenbasis']).

fuege_hinzu(r(X,Y)) :-
   faktum(r(X,Y),empirisch),
   schreibe_liste([r(X,Y),' bereits in
   der Datenbasis vorhanden.']),!.

fuege_hinzu(r(X,Y)) :-
   knoten(M),
   (
   not member_1(X,M),
   schreibe_liste([X,' ist kein
   vorhandenes Element'])
   ;
   not member_1(Y,M),
   schreibe_liste([Y,' ist kein
   vorhandenes Element'])
   ),!.

fuege_hinzu(r(X,Y)) :-
   entferne(alle_schluesse),
   assertz(faktum(r(X,Y),empirisch)).

fuege_hinzu_l(_,[]).
fuege_hinzu_l(El,[El2|Rest]) :-
   add_corr_rel(r(El,El2)),
   fuege_hinzu_l(El,Rest).

add_corr_rel(r(X,Y)) :-
   assertz(faktum(r(X,Y),korrigiert)),
   schreibe_liste(['Fuege hinzu: ',
   r(X,Y)]).



/*  INFERENZ-MASCHINE  */


zeige(X) :-
   check(X),
   faktum(X,_).

zeige(X) :-
   regel(Nr,_,X,_),
   schreibe_liste([X,' gilt']).

leite_ab(X) :-
   tron(nein),
   check(X),
   faktum(X,_).

leite_ab(X) :-
   tron(nein),
   regel(Nr,_,X,Ursach),
   not schon_vorhanden(X),
   schreibe_liste(['Mit Regel ',Nr,'
   abgeleitet: ', X,' ->  Faktenbasis']),
   asserta(faktum(X,[Nr|Ursach])).

leite_ab(X) :-
   tron(ja),
   check(X),
   schreibe_liste(['Pruefe Faktenbasis:
   ',X]),
   faktum(X,_),
   schreibe_liste([X,' vorhanden']).

leite_ab(X) :-
   tron(ja),
   schreibe_liste(['Nicht vorhanden:    ',
   X]),
   clause(regel(Nr,_,X,Ursach),U),
   schreibe_liste(['Versuche Regel:     ',
   Nr]),nl,
   gib_regel_aus(Nr,_),nl,
   regel(Nr,_,X,Ursach),
   not schon_vorhanden(X),
   schreibe_liste(['Regel ',Nr,'
   erfolgreich.']),
   schreibe_liste(['
   Praemissen:']),
   schreibe_liste(Ursach),
   schreibe_liste([X,'    ->
   Faktenbasis']),
   asserta(faktum(X,[Nr|Ursach])).

check(n(X,Y)) :-
   (
   var(X)
   ;
   var(Y)
   ),
   knoten(O),
   behandle_paare(O,O),!.

check(_).

schon_vorhanden(clique(X)) :-
   findall(Z,faktum(clique(Z),_),L),
   test_auf_gleiche_menge(X,L),
   schreibe_liste([X,'
   schon vorhanden.']),nl.

schon_vorhanden(m_clique(X)) :-
   findall(Z,faktum(m_clique(Z),_),L),
   test_auf_gleiche_menge(X,L),
   schreibe_liste([X,' schon
   vorhanden.']),nl.

schon_vorhanden(triadentyp(X,Y,_)) :-
   findall(Z,faktum(triadentyp(Z,Y,_),_),
   L),
   test_auf_gleiche_menge(X,L),
   schreibe_liste([X,' schon
   vorhanden.']),nl.

schon_vorhanden(X) :- faktum(X,_).

test_auf_gleiche_menge(_,[]) :-
   !,fail.
test_auf_gleiche_menge(X,[K|Rest]) :-
   gleiche_menge(X,K),!.
test_auf_gleiche_menge(X,[_|Rest]) :-
   test_auf_gleiche_menge(X,Rest).

behandle_paare([],_).
behandle_paare([K|Rest],L) :-
   behandle_paare2(K,L),!,
   behandle_paare(Rest,L).

behandle_paare2(_,[]).
behandle_paare2(El,[El|R])  :-
   behandle_paare2(El,R).
behandle_paare2(El,[K|R])  :-
   leite_ab(n(El,K)),
   behandle_paare2(El,R).

behandle_paare2(El,[_|R]) :-
   behandle_paare2(El,R).
```

```
/*  THEORETISCHER TEIL VON HLT  */


regel(1,man, m(X,Y),[r(X,Y),r(Y,X)]) :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,X)).

regel(2,man, a(X,Y),[r(X,Y), not,
   r(Y,X)])
   :-
   leite_ab(r(X,Y)),
   not leite_ab(r(Y,X)).

regel(3,man, n(X,Y),[not, r(X,Y),
   not,r(Y,X)])
   :-
   not leite_ab(r(X,Y)),
   not leite_ab(r(Y,X)),
   ungleich(X,Y).

regel(4,graph,graph(t_graph,false),
   [r(X,Y), r(Y,Z), not, r(X,Z)])
   :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,Z)),
   ungleich(X,Z),
   not leite_ab(r(X,Z)).

regel(5,graph,graph(t_graph,true),
   [r(x,y),r(y,z),r(x,z), 'fuer alle x,
    y, z'])
   :-
   not leite_ab(graph(t_graph,false)).

regel(6,transtest,triade([X,Y,Z],
   intransitiv), [r(X,Y),r(Y,Z), not,
   r(X,Z)])
   :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,Z)),
   ungleich(X,Z),
   not leite_ab(r(X,Z)).

regel(7,transtest,triade([X,Y,Z],
   transitiv),[r(X,Y),r(Y,Z),r(X,Z)])
   :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,Z)),
   leite_ab(r(X,Z)).

regel(8,cliquen,m_clique([X1|L]),
   [m(x,y),' fuer alle x y aus', [X1|L]])
   :-
   leite_ab(graph(t_graph,true)),
   knoten(X),
   member(X1,X),
   findall(Y,leite_ab(m(X1,Y)),L),
   not L = [].

regel(9,cliquen,a_stern(U,V),[a(x,y),
   ' fuer alle x aus', U,' und y
   aus ',V])
   :-
   leite_ab(m_clique(U)),
   leite_ab(m_clique(V)),
   ungleich(U,V),
   sind_alle_a_waehler_von(U,V).

regel(10,triadentyp,triadentyp([X,Y,Z],
   '201',intransitiv),
   [m(X,Z),m(X,Y),n(Y,Z)])
   :-
   leite_ab(m(X,Z)),
   leite_ab(m(X,Y)),
   leite_ab(n(Y,Z)),
   ungleich(X,Y,Z).

regel(11,triadentyp,triadentyp([X,Y,Z],
   '210',intransitiv),[m(X,Y),m(X,Z),
   a(Y,Z)])
   :-
   leite_ab(m(X,Y)),
   leite_ab(m(X,Z)),
   leite_ab(a(Y,Z)),
   ungleich(X,Y,Z).

regel(12,triadentyp,triadentyp([X,Y,Z],
   '120c',intransitiv),[m(X,Z),a(X,Y),
   a(Y,Z)])
   :-
   leite_ab(m(X,Z)),
   leite_ab(a(X,Y)),
   leite_ab(a(Y,Z)),
   ungleich(X,Y,Z).

regel(13,triadentyp,triadentyp([X,Y,Z],
   '030c',intransitiv),[a(Z,X),a(X,Y),
    a(Y,Z)])
   :-
   leite_ab(a(Z,X)),
   leite_ab(a(X,Y)),
   leite_ab(a(Y,Z)),
   ungleich(X,Y,Z).

regel(14,triadentyp,triadentyp([X,Y,Z],
   '111d', intransitiv),[m(X,Z),a(Y,Z),
   n(X,Y)])
   :-
   leite_ab(m(X,Z)),
   leite_ab(a(Y,Z)),
   leite_ab(n(X,Y)),
   ungleich(X,Y,Z).

regel(15,triadentyp,triadentyp([X,Y,Z],
   '111u',intransitiv),[m(X,Z),a(Z,Y),
   n(X,Z)])
   :-
   leite_ab(m(X,Z)),
   leite_ab(a(Z,Y)),
   leite_ab(n(X,Y)),
   ungleich(X,Y,Z).

regel(16,triadentyp,triadentyp([X,Y,Z],
   '021c',intransitiv),[a(X,Y),a(Y,Z),
   n(X,Z)])
   :-
   leite_ab(a(X,Y)),
   leite_ab(a(Y,Z)),
   leite_ab(n(X,Z)),
   ungleich(X,Y,Z).

regel(17,triadentyp,triadentyp([X,Y,Z],
   '003',leer_transitiv),[n(X,Y),n(X,Z),
   n(Y,Z)])
   :-
   leite_ab(n(X,Y)),
   leite_ab(n(X,Z)),
   ungleich(Y,Z),
   leite_ab(n(Y,Z)),
   ungleich(X,Y,Z).

regel(18,triadentyp,triadentyp([X,Y,Z],
   '102',leer_transitv),[m(X,Y),n(X,Z),
   n(Z,Y)])
   :-
   leite_ab(m(X,Y)),
   leite_ab(n(X,Z)),
   leite_ab(n(Z,Y)),
   ungleich(X,Y,Z).
```

```
regel(19,triadentyp,triadentyp([X,Y,Z],
   '021d',leer_transitiv),[a(Y,X),a(Y,Z),
   n(X,Z)])
   :-
   leite_ab(a(Y,X)),
   leite_ab(a(Y,Z)),
   leite_ab(n(X,Z)),
   ungleich(X,Y,Z).

regel(20,triadentyp,triadentyp([X,Y,Z],
   '021u',leer_transitiv),[a(X,Z),a(Y,Z),
   n(X,Y)])
   :-
   leite_ab(a(X,Z)),
   leite_ab(a(Y,Z)),
   leite_ab(n(X,Y)),
   ungleich(X,Y,Z).

regel(21,triadentyp,triadentyp([X,Y,Z],
   '012',leer_transitiv),[a(X,Y),n(Y,Z),
   n(X,Z)])
   :-
   leite_ab(a(X,Y)),
   leite_ab(n(Y,Z)),
   leite_ab(n(X,Z)),
   ungleich(X,Y,Z).

regel(22,triadentyp,triadentyp([X,Y,Z],
   '300',transitiv),[m(X,Y),m(X,Z),
   m(Y,Z)])
   :-
   leite_ab(m(X,Y)),
   leite_ab(m(X,Z)),
   leite_ab(m(Y,Z)).

regel(23,triadentyp,triadentyp([X,Y,Z],
   '120u',transitiv),[m(X,Z),a(Y,Z),
   a(X,Z)])
   :-
   leite_ab(m(X,Z)),
   leite_ab(a(X,Y)),
   leite_ab(a(Z,Y)),
   ungleich(X,Y,Z).

regel(24,triadentyp,triadentyp([X,Y,Z],
   '120d',transitiv),[m(X,Z),a(Y,X),
   a(Y,Z)])
   :-
   leite_ab(m(X,Z)),
   leite_ab(a(Y,X)),
   leite_ab(a(Y,Z)),
   ungleich(X,Y,Z).

regel(25,triadentyp,triadentyp([X,Y,Z],
   '030t',transitiv),[a(X,Z),a(X,Y),
   a(Z,Y)])
   :-
   leite_ab(a(X,Z)),
   leite_ab(a(X,Y)),
   leite_ab(a(Z,Y)),
   ungleich(X,Y,Z).

regel(26,graph,
   graph(completely_connected,true),
   [graph(t_graph,true),'es gibt m(x,y):
   ',m(X,Y),' es gibt kein n(x,y)',' es
   gibt kein a(x,y)'])
   :-
   leite_ab(graph(t_graph,true)),
   leite_ab(m(X,Y)),
   not leite_ab(n(_,_)),
   not leite_ab(a(_,_)).

regel(27,graph,
   graph(transitive_tournament,true),
   [graph(t_graph,true),' es gibt ein
   a(x,y): ',a(X,Y),' es gibt kein
   m(x,y)',' es gibt kein n(x,y)'])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(m(_,_)),
   not leite_ab(n(_,_)),
   leite_ab(a(X,Y)).

regel(28,graph,
   graph(completely_disconnected,true),
   [graph(t_graph,true),' es gibt ein
   n(x,y): ',n(X,Y),' es gibt kein
   m(x,y)',' es gibt kein a(x,y)'])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(m(_,_)),
   leite_ab(n(X,Y)),
   not leite_ab(a(_,_)).

regel(29,graph,graph(quasi_series,true),
   [graph(t_graph,true),' es gibt ein
   m(x,y): ',m(X1,X2),' es gibt kein
   n(x,y)',' es gibt ein a(x,y): ',
   a(Y1,Y2)])
   :-
   leite_ab(graph(t_graph,true)),
   leite_ab(m(X1,X2)),
   not leite_ab(n(_,_)),
   leite_ab(a(Y1,Y2)).

regel(30,graph,graph(partial_order,true),
   [graph(t_graph,true),' es gibt kein
   m(x,y)',' es gibt ein n(x,y):
   ',n(X1,X2),' es gibt ein a(x,y): ',
   a(Y1,Y2)])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(m(_,_)),
   leite_ab(n(X1,X2)),
   leite_ab(a(Y1,Y2)).

regel(31,graph,graph(clusterable_graph,
   true),[graph(t_graph,true),'es gibt
   m(X,Y): ',m(X1,X2),' es gibt
   n(X,Y): ',n(Y1,Y2),' es gibt kein
   a(X,Y)'])
   :-
   leite_ab(graph(t_graph,true)),
   leite_ab(m(X1,X2)),
   leite_ab(n(Y1,Y2)),
   not leite_ab(a(_,_)).

regel(32,graph,graph(structural_balance,
   true),[graph(t_graph,true),
   'es gibt nur Triaden 102, 300',
   'es gibt keine Triaden 003, 012, 021u,
   021d, 030t'])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(triadentyp(_,'003',_)),
   not leite_ab(triadentyp(_,'012',_)),
   not leite_ab(triadentyp(_,'021u',_)),
   not leite_ab(triadentyp(_,'021d',_)),
   not leite_ab(triadentyp(_,'030t',_)),
   not leite_ab(triadentyp(_,'120u',_)),
   not leite_ab(triadentyp(_,'120d',_)).

regel(33,graph,graph(ranked_clusters,
   true),[graph(t_graph,true),'es gibt
   keine 012-Triaden'])
   :-
   leite_ab(graph(t_graph,true)),
   not leite_ab(triadentyp(_,'012',_)).
```

```
regel(34,cliquen,clique(Z),
   ['m(x,y) fuer alle x,y aus ',Z,
   ' nicht es gibt ein z aus ', Rest,
   ' so dass: m(z,x) fuer alle x aus ',
   Z])
   :-
   knoten(X),
   member(X1,X),
   zyklus(X1,Z,m),
   sind_alle_m_verbunden(Z),
   differenz(X,Z,Rest),
   nicht_m_verbunden(Rest,Z).

regel(35,cliquen,status(Clique,Status,
   Groesse),[Clique, ' mit ', Groesse, '
   Mitgliedern wird gewaehlt von ',
   Status, ' Aussenelementen: ',
   Waehlerliste])
   :-
   leite_ab(clique(Clique)),
   finde_alle_waehler_von(Clique,Clique,
   Waehlerliste),
   card(Waehlerliste,Status),
   card(Clique,Groesse).

regel(36,graph,balancegrad(Balance),
   ['Anzahl intransitiver Tripel: ',
   'Anzahl aller Tripel: ',Alle_Tripel])
   :-
   zaehle(triade(_,intransitiv),In_zahl),
   knoten(X),
   zaehle(X,Kn_zahl),
   Alle_Tripel is
   Kn_zahl * (Kn_zahl-1) * (Kn_zahl-2)/6,
   Balance is 1 - In_zahl*1/Alle_Tripel.

regel(37,man, m_schlange(X,Y),
   ['es gibt einen m-weg ',W,' von ',X,
   ' nach ',Y])
   :-
   knoten(K),
   member(X,K),
   member(Y,K),
   not X = Y,
   pfad(X,Y,W,m).



/* Hilfspraedikate zu den Regeln */


sind_alle_a_waehler_von([],_).
sind_alle_a_waehler_von([K|Rest],V) :-
   ist_ein_a_waehler_von(K,V),
   sind_alle_a_waehler_von(Rest,V).

ist_ein_a_waehler_von(_,[]).
ist_ein_a_waehler_von(K,[K2|Rest]) :-
   leite_ab(a(K,K2)),
   ist_ein_a_waehler_von(K,Rest).

nicht_m_verbunden([],_).
nicht_m_verbunden([X|R],Pfad) :-
   not ist_m_verbunden(X,Pfad),
   nicht_m_verbunden(R,Pfad).

ist_m_verbunden(_,[]).
ist_m_verbunden(X,[Y|R]) :-
   leite_ab(m(X,Y)),
   ist_m_verbunden(X,R).

sind_alle_m_verbunden([]).
sind_alle_m_verbunden([X|R]) :-
   ist_m_verbunden(X,R), %write(R),
   sind_alle_m_verbunden(R).

finde_alle_waehler_von([],_,[]).

finde_alle_waehler_von([C|Rest],Cl,
   Neuwahl)
   :-
   finde_alle_waehler_von(Rest,Cl,
   Wahllist),
   finde_waehler_von(C,Cl,W),
   append(W,Wahllist,Neuwahl).

finde_waehler_von(X,Cl,L) :-
   findall(Y,leite_ab(r(Y,X)),L),
   not member(Y,Cl).



/*  GENERIERUNG VON BALANCE -  */
/*     KORREKTUR DER DATEN     */


/* 1. Korrekturverfahren */

balance1 :-
   entferne(alle_schluesse),
   zeige(triade([A,B,C],intransitiv)),
   schreibe_liste_nl([r(A,B),r(B,C),not,
   r(A,C)]),
   add_corr_rel(r(A,C)),
   fail.

balance1 :-
   write('Fertig\n').


/* 2. Korrekturverfahren */

balance2 :-
   write('Cliquen- und Cliquenstatus
   muessen vollstaendig abgeleitet
   sein'),nl,
   write('Ist dies der Fall?'),
   read(Ch),!,
   parameter_wahl(Typ,Ordnung),
   erzeuge_order_list(Typ,Ordnung,
   Cliquelist),!,
   knoten(X),
   entferne(alle_schluesse),
   plaette(Cliquelist,L1),
   differenz(X,L1,L),
   durchsuche_clique(L1,L1,[]),
   durchsuche_aussen(L,L).

erzeuge_order_list(status,Ordnung,
   Sortliste)
   :-
   findall([M,Stat],
   faktum(status(M,Stat,_),_),Statliste),
   insertsort(Statliste,Sortliste,
   Ordnung).

erzeuge_order_list(groesse,Ordnung,
   Sortliste)
   :-
   findall([M,Card],
   faktum(status(M,_,Card),_),
   Statliste),
   insertsort(Statliste,Sortliste,
   Ordnung).

durchsuche_aussen([],_).
durchsuche_aussen([K|Rest],L) :-
   schreibe_liste(['Erstelement
   Aussen: ', K,  Rest ',Rest]),
   korrigiere_wahl_von(K),!,
   durchsuche_aussen(Rest,L).
durchsuche_aussen([Element|Rest],L) :-
   durchsuche_aussen(Rest,L).
```

```
test(X,Z) :-
   leite_ab(r(X,Y)),
   leite_ab(r(Y,Z)),
   not (X=Y),
   not (X=Z),
   not (Y=Z),
   not faktum(r(X,Z),_).

durchsuche_clique([],_,_).
durchsuche_clique([Element|Rest],L,L2) :-
   member(Element,L2),!,
   durchsuche_clique(Rest,L,L2).
durchsuche_clique([Element|Rest],L,L2)
   :-
   schreibe_liste(['Erstelement: ',
   Element,' Rest ',Rest]),
   test(Element,Z),
   korrigiere_wahl_von(Element),
   finde_waehler_von(Element,L,L1),
   append(Rest,L1,Neulist),!,
   durchsuche_clique(Neulist,L,
   [Element|L2]).
durchsuche_clique([Element|Rest],L,L2)
   :-
   finde_waehler_von(Element,L,L1),
   append(Rest,L1,Neulist),!,
   durchsuche_clique(Neulist,L,
   [Element|L2]).

finde_waehler_von(X,V_Liste, Erg) :-

findall(Y,sortiere_aus(X,Y,V_Liste),L),
   entferne_mehrfach(L,Erg).

sortiere_aus(X,Y,L) :-
   faktum(r(Y,X),_),
   not member(Y,L).

korrigiere_wahl_von(X) :-
   findall(Kand,test(X,Kand),L),
   entferne_mehrfach(L,Neu),
   add_neu(X,Neu).

add_neu(X,[]).
add_neu(X,[K|R]) :-
   tiefe(X,K),
   add_neu(X,R).
add_neu(X,[_|R]) :-
   add_neu(X,R).

tiefe(X,El) :-
   test(X,El),
   add_corr_rel(r(X,El)),
   tiefe(X,Z).


/* 3. Korrekturverfahren */

balance3 :-
   write('Cliquen- und Cliquenstatus muss
   vollstaendig abgeleitet sein'),nl,
   write('Ist dies der Fall? '),read(X),
   parameter_wahl(Typ,Ordnung),
   erzeuge_order_list(Typ,Ordnung,
   Sortliste),
   suche_cliquen_paar(Sortliste),
   balance2.

suche_cliquen_paar([]).
suche_cliquen_paar([[M1,Stat1]|Rest]) :-
   faktum(status(M1,Stat1,_),_),
   faktum(status(M2,Stat2,_),_),
   ungleich(M1,M2),
   Stat1 >= Stat2,
   suche_zyklus(M1,M2),
   suche_cliquen_paar(Rest).

suche_cliquen_paar([_|Rest]):-
   suche_cliquen_paar(Rest).

suche_zyklus([X1|M1],[X2|M2]) :-
   einbahn_pfad(X1,X2,Pfad,r),
   schreibe_liste(['Pfad von ',X1,
   ' nach ',X2 , ' gefunden:',Pfad]),
   einbahn_pfad(X2,X1,Pfad2,r),
   schreibe_liste(['Pfad von ',X2,
   ' nach ',X1 ,' gefunden:',Pfad2]),
   finde_cliquen_uebergang(Pfad,[X1|M1],
   Rel),
   write('Cliquenuebergang: '),
   write(Rel),nl,
   entferne_relation(Rel),
   schreibe_liste([Rel,' geloescht ',
   ' Clique']),
   suche_zyklus([X1|M1],[X2|M2]).

entferne_relation(Rel) :-
   retract(faktum(Rel,_)).

finde_cliquen_uebergang([],_,_).

finde_cliquen_uebergang([X,Y|Rest],
   Clique,r(X,Y))
   :-
   member(X,Clique),
   not member(Y,Clique).

finde_cliquen_uebergang([X,Y|Rest],
   Clique,Z)
   :-
   finde_cliquen_uebergang([Y|Rest],
   Clique,Z).

zyklus(X,Pfad,Rel_typ) :-
   R=..[Rel_typ,Y,X],
   leite_ab(R),
   pfad(X,Y,Pfad,Rel_typ).

pfad(X,Y,Pfad,P) :-
   pfad1(X,[Y],Pfad,P).

pfad1(X,[X|Pfad1],[X|Pfad1],P).

pfad1(X,[Y|Pfad1],Pfad,P) :-
   R=..[P,Y1,Y],
   leite_ab(R),
   not member(Y1,Pfad1),
   pfad1(X,[Y1,Y|Pfad1],Pfad,P).

einbahn_pfad(X,Y,Pf,R) :-
   pfad(X,Y,Pf,R),!.


/*  ERKLAERUNGSKOMPONENTE  */


tron(nein).

t_ein :-
   retract(tron(_)),
   asserta(tron(ja)).
t_aus :-
   retract(tron(_)),
   asserta(tron(nein)),
   abolish(angewandte_regel(_,_)).

wie :-
   faktum(X,_),
   wie(X).
```

```
wie(X) :-
   faktum(X,[Nr|Rest]),
   schreibe_liste(Rest),
   gib_regel_aus(Nr,_),
   write('Also: '),
   write(X).

regel(X) :-
   numeric(X),
   gib_regel_aus(X,_).
regel(X) :-
   not numeric(X),
   regel_gruppe(Nr,X).

regeln :-
   alle_regeln(1).

alle_regeln(Nr) :-
   gib_regel_aus(Nr,_),
   N1 is Nr + 1,
   alle_regeln(N1).
alle_regeln(_) :-
   write('\nAll rules').

regel_gruppe(Nr,Typ) :-
   gib_regel_aus(Nr,Typ),
   N1 is Nr +1,
   regel_gruppe(N1,Typ).
regel_gruppe(Nr,Typ) :-
   Nr < 50,
   N1 is Nr + 1,
   regel_gruppe(N1,Typ).
regel_gruppe(_,_).

gib_regel_aus(Nr,Typ) :-
   clause(regel(Nr,Typ,Konkl,Beding),
   Ursach),
   asserta(schon_vergeben(dummy)),
   schreibe_liste(['Regelnr.',Nr]),
   write('WENN '),
   schreibe_ursachen(Ursach),
   write(' DANN '),
   schreibe_konklusion(Konkl),
   abolish(schon_vergeben(_)),
   nl,nl,!.

instantiiere(X) :-
   X=..L,
   pruefe_instanz(L,L1),
   X=..L1.

pruefe_instanz([],[]).
pruefe_instanz([X|R],[X|R2]) :-
   var(X),
   variablenliste(L),
   member(V,L),
   not schon_vergeben(V),
   X = V,
   asserta(schon_vergeben(V)), !,
   pruefe_instanz(R,R2).

pruefe_instanz([X|R],[X|R2]) :-
   nonvar(X),
   pruefe_instanz(R,R2).

schreibe_konklusion(Konkl) :-
   instantiiere(Konkl),
   write(Konkl).

schreibe_ursachen(leite_ab(R)) :-
   instantiiere(R),
   write(R).

schreibe_ursachen(not leite_ab(R)) :-
   write(' NOT '),
   instantiiere(R),
   write(R).

schreibe_ursachen(leite_ab(R) ','
   Ursachen)
   :-
   instantiiere(R),
   write(R),
   write(' UND '),
   schreibe_ursachen(Ursachen).

schreibe_ursachen(not leite_ab(R) ','
   Ursachen)
   :-
   write(' NOT '),
   instantiiere(R),
   write(R),
   write(' UND '),
   schreibe_ursachen(Ursachen).

schreibe_ursachen(ungleich(_,_) ','
   Ursachen)
   :-
   schreibe_ursachen(Ursachen).

schreibe_ursachen(ungleich(_,_)).

schreibe_ursachen(ungleich(_,_,_) ','
   Ursachen)
   :-
   schreibe_ursachen(Ursachen).

schreibe_ursachen(ungleich(_,_,_)).

schreibe_ursachen(Rel ',' Ursachen) :-
   instantiiere(Rel),
   write(Rel),
   write(' UND '),
   schreibe_ursachen(Ursachen).

schreibe_ursachen(Rel) :-
   instantiiere(Rel),
   write(Rel).



/*  SCHNITTSTELLE ZUM BENUTZER  */

hilfe :-
   nl,nl,
   write('***** HOLLAND-LEINHARDT-
          THEORIE (HLT 1971) *****
         *'),nl,nl,
   write('Datenbasis aendern (lesen,
          loeschen, etc.
          -> hilfe_db.'),nl,
   write('Fakten deduzieren
          -> hilfe_dedukt.'),nl,
   write('Erklaerungen
          -> hilfe_erklaer.'),nl,
   write('Balance-Herstellung
          -> hilfe_balance.'),nl.

hilfe_db :-
   write('Moegliche Eingaben:'),nl,nl,
   write('DATENBASIS AENDERN'),nl,
   write('- Daten eingeben
          -> lesen.'), nl,
   write('- Alte Wissensbasis voellig
          loeschen
          -> neu.'),nl,
   write('- Einzelnen Fakt x loeschen
          -> entferne(x).'),nl,
   write('- Nur Schlussfolgerungen
          loeschen
          -> entferne(alle_schluesse).'),
          nl,
   write('- Korrigierte Daten entfernen
          -> entferne(korr_daten).'),nl,
   write('- Einzelnen Fakt entfernen
          -> entferne(fakt).'),nl,
```

```
   write('- Einzelnen Fakt x hinzunehmen
          -> fuege_hinzu(x).'),nl.

hilfe_dedukt :-
   write('Moegliche Eingaben:'),nl,nl,
   write('FAKTEN UND SCHLUESSE'),nl,
   write('- Einzelnen Fakt x herleiten
          -> leite_ab(x)'),nl,
   write('- M- A- oder N-Relationen
          -> m, a oder n.'),nl,
   write('- Alle Relationen
          -> man.'),nl,
   write('- Alle M-Cliquen
          -> m_clique'),nl,
   write('- A*-Relation
          -> a_stern'),nl,
   write('- T-Graph pruefen
          -> t_graph'),nl,
   write('- Intransitive Triaden
          -> trans_test'),nl,
   write('- Alle Spezialmodelle
           -> graph'),nl,
   write('- Balancegrad
           -> balancegrad'),nl,
   write('- Alle Cliquen
           -> clique'),nl,
   write('- Status von allen Cliquen
           -> status'),nl,nl,
   write('- Ueberblick ueber alle
           gueltige Fakten
           -> fakten.'),nl.

hilfe_erklaer :-
   write('Moegliche Eingaben:'),nl,nl,
   write('- Schlussrechtfertigung fuer
            Fakt:
           -> wie(fakt).'),nl,
   write('- Schlussrechtfertigung fuer
           letzten Fakt
           -> wie.'),nl,
   write('- Alle Regeln ausgeben
           -> regeln'),nl,
   write('- Regelnummer x ausgeben
           -> regel(x)'),nl,
   write('- Bestimmte Regeltypen typ
           ausgeben
           -> regel(typ)'),nl,
   write('- Trace ein-/ausschalten
           -> t_ein/t_aus'),nl.

hilfe_balance :-
   write('Moegliche Eingaben:'),nl,nl,
   write('- 1. Balance-Generierungs-
           Verfahren
           -> balance1.'),nl,
   write('- 2. Balance-Generierungs-
           Verfahren
           -> balance2.'),nl,
   write('- 3. Balance-Generierungs-
           Verfahren
           -> balance3.'),nl.



/*  HILFS-PRAEDIKATE ZUM ABLEITEN  */

m :-
   leite_ab(m(X,Y)),
   fail.
m.

n :-
   leite_ab(n(X,Y)),
   fail.
n.

a :-
   leite_ab(a(X,Y)),
   fail.
a.

man :-
   m,
   n,
   a.
man.

m_clique :-
   leite_ab(m_clique(X)),
   fail.
m_clique.

clique :-
   leite_ab(clique(X)),
   fail.
clique.

t_graph :-
   leite_ab(graph(t_graph,X)),
   fail.
t_graph.

a_stern :-
   leite_ab(a_stern(X,Y)),
   fail.
a_stern.

graph :-
   leite_ab(graph(X,Y)),
   fail.
graph.

status :-
   leite_ab(status(X,Y,Z)),
   fail.
status.

trans_test :-
   leite_ab(triade(X,intransitiv)),
   fail.
trans_test.

balancegrad :-
   leite_ab(balancegrad(X)).

m_schlange :-
   leite_ab(m_schlange(X,Y)),
   fail.
m_schlange.


/*  FAKTEN-UEBERBLICK  */

fakten :-
   grundfakt,!,
   high_level_fakt.

grundfakt :-
   elemente,
   r_rel,
   m_rel,
   a_rel,
   n_rel.

high_level_fakt :-
   faktum(X,_),
   not fakt_schon_gezeigt(X),
   write(X),nl,
   fail.
high_level_fakt :- nl.

fakt_schon_gezeigt(r(_,_)).
fakt_schon_gezeigt(m(_,_)).
fakt_schon_gezeigt(a(_,_)).
```

```
fakt_schon_gezeigt(n(_,_)).


elemente :-
   write('Knoten: '),
   knoten(X),
   write(X),
   nl,nl.

r_rel :-
   findall([X,Y],faktum(r(X,Y),_),Liste),
   schreibe_liste(['R-Relationen
   -> ',Liste]),
   nl.

m_rel :-
   findall([X,Y],faktum(m(X,Y),_),Liste),
   schreibe_liste(['M-Relationen
   -> ',Liste]),
   nl.

a_rel :-
   findall([X,Y],faktum(a(X,Y),_),Liste),
   schreibe_liste(['A-Relationen
   -> ',Liste]),
   nl.

n_rel :-
   findall([X,Y],faktum(n(X,Y),_),Liste),
   schreibe_liste(['N-Relationen
   -> ',Liste]),
   nl.

korr_rel :-
   findall([X,Y],
   faktum(r(X,Y),korrigiert),Liste),
   schreibe_liste(['Korrigierte
   Relationen -> ',Liste]),
   nl.



/*  MENGENTHEORETISCHE HILFSPRAEDIKATE */


/* Kreuzprodukt bilden */

cross(X,Y,Z) :-
   listall(K,pair(X,Y,K),Z1),
   mkset(Z1,Z).

 cross(X,Y,Z) :-
   send_msg(cross,sup,fail,cross(X,Y,Z)).

send_msg(FromWhom,sup,fail,Msg) :-
   assertz(msg(FromWhom,sup,fail,Msg)).
send_msg(FromWhom,sup,failfail,Msg) :-
   assertz(msg(FromWhom,sup,fail,Msg)),
   !,fail .
send_msg(FromWhom,sup,Type,Msg) :-
   assertz(msg(FromWhom,sup,Type,Msg)).

listall(X,G,_) :-
   asserta(found(mark)),
   call(G),
   asserta(found(X)),
   fail.
listall(_,_,L) :-
   collect([],M),
   !, L = M.

collect(S,L) :-
   collget(X),
   !, collect([X|S],L).
collect(L,L).

collget(X) :-
   retract(found(X)),
   !, X \= mark.

pair(X,Y,K) :- member(A,X),
   member(B,Y),
   K = [A,B].

mkset([],[]).
mkset([X|L],Z) :-
   member(X,L),!,
   mkset(L,Z).
mkset([X|L],[X|Z]) :-
   mkset(L,Z).

/* Elementschaftsrelation (nur einmal
erfuellbar) */

member_1(Element,[Element|_]) :- !.
member_1(Element,[_|Rest]) :-
member_1(Element,Rest).

/* Argument ist eine Liste */

is_list([]).
is_list([_|_]).

/* Listenelemente loeschen */

delete([],L,L).
delete([K|Rest],In,Erg) :-
   del_element(K,In,Out),
   delete(Rest,Out,Erg).

del_element(El,[El|Rest],Rest).
del_element(El,[K|Rest],[K|Erg]) :-
   del_element(El,Rest,Erg).

/* Differenzmenge */

differenz(L,[],L) :- !.
differenz([Kopf|Rest],L,U) :-
   member(Kopf,L),!,
   differenz(Rest,L,U).
differenz([Kopf|Rest],L,[Kopf|Rest2]) :-
   !,
   differenz(Rest,L,Rest2).
differenz(_,_,[]).

/* Aus Liste Menge bilden (Mehrfache
Elemente entfernen) */

entferne_mehrfach([],[]).
entferne_mehrfach([K|R],M) :-
   entferne_mehrfach(R,M),
   member(K,M).
entferne_mehrfach([K|R],[K|R2]) :-
   entferne_mehrfach(R,R2),
   not member(K,R2).

/* Durchschnittsmenge */

durchschnitt([],X,[]).
durchschnitt([K|R],Y,[K|Z]) :-
   member(K,Y),!,
   durchschnitt(R,Y,Z).
durchschnitt([K|R],Y,Z) :-
   durchschnitt(R,Y,Z).

/* Teilmenge */

teilmenge([],Y).
teilmenge([K|X],Y) :-
   member(K,Y),
   teilmenge(X,Y).
```

```
/* Gleiche Menge */

gleiche_menge(X,Y) :-
   teilmenge(X,Y),
   teilmenge(Y,X).

/* Liste plaetten */

plaette([],[]) :- !.
plaette([[]|R],L) :-
   !,plaette(R,L).
plaette([[K|K1]|R],L) :-
   !,plaette([K|K1],L1),
   plaette(R,L2),
   append(L1,L2,L).
plaette([K|R],L) :-
   integer(K),!,
   plaette(R,L).
plaette([K|R],[K|L]) :-
   plaette(R,L).



/*  ALLGEMEINE HILFSPRAEDIKATE  */


ungleich(A,B) :-
   not A=B.
ungleich(A,B,C) :-
   not A=B,
   not A=C,
   not B=C.

card([],0).
card([K|Rest],N) :-
   card(Rest,N1),
   N is N1 + 1.

insertsort([],[],Typ).
insertsort([X|Rest],Sortiert,Typ) :-
   insertsort(Rest,Sortierter_Rest,Typ),
   fuege_ein(X,Sortierter_Rest,Sortiert,
   Typ).
fuege_ein(X,[Y|Sortiert],[Y|Sortiert1],
   Typ)
   :-
   ord_rel(X,Y,Typ),!,
   fuege_ein(X,Sortiert,Sortiert1,Typ).
fuege_ein(X,Sortiert,[X|Sortiert],Typ).

ord_rel(X,Y,ab) :-
   integer(X),
   integer(Y),
   X =< Y.

ord_rel(X,Y,auf) :-
   integer(X),
   integer(Y),
   X >= Y.

ord_rel([Atom1,Zahl1],[Atom2,Zahl2],Typ)
   :-
   integer(Zahl1),
   integer(Zahl2),
   ord_rel(Zahl1,Zahl2,Typ).

zaehle(X,Z1) :-
   is_list(X),
   card(X,Z1).

zaehle(X,Z1) :-
   not is_list(X),
   asserta(anzahl(X,0)),
   leite_ab(X),
   retract(anzahl(_,Z)),
   Z1 is Z + 1,
   asserta(anzahl(X,Z1)),
   fail.

zaehle(X,Z1) :-
   not is_list(X),
   anzahl(X,Z1),
   retract(anzahl(_,_)),!.

bereinige([],[]).
bereinige([A|Rest],Erg) :-
   trash(Liste),
   member_1(A,Liste),
   bereinige(Rest,Erg).
bereinige([A|Rest],[A|Erg]) :-
   bereinige(Rest,Erg).

variablenliste([x,y,z,u,v,w,x1,x2,x3,x4,
                x5,x6,x7]).

trash(['.',',',';',':','?','!']).

schreibe_liste([]) :- nl,!.
schreibe_liste([Kopf|Rest]) :-
   write(Kopf),
   tab(1),
   schreibe_liste(Rest).

schreibe_liste_nl(L) :-
   schreibe_liste(L),nl.

parameter_wahl(Typ,Ordnung) :-
   nonvar(Typ),
   nonvar(Ordnung),
   member(Typ,[status,groesse]),
   member(Ordnung,[auf,ab]),!.

parameter_wahl(Typ,Ordnung) :-
   write('\nParameter CLIQUEN-STATUS\n'),
   write('       (1) Reihenfolge von
                     GERINGEM zu GROSSEN
                     Status\n'),
   write('       (2) Reihenfolge von
                     GROSSEM zu GERINGEM
                     Status\n'),
   write('\nParameter CLIQUEN-
            GROESSE\n'),
   write('       (3) Reihenfolge von
                     KLEINEN zu GROSSEN
                     Cliquen\n'),
   write('       (4) Reihenfolge von
                     GROSSEN zu KLEINEN
                     Cliquen\n\n'),
   write('Wahl ==> '),
   zeichen_lesen(C),
   (
   C == '1',
   Typ=status,
   Ordnung=auf,!
   ;
   C == '2',
   Typ=status,
   Ordnung=ab,!
   ;
   C == '3',
   Typ=groesse,
   Ordnung=auf,!
   ;
   C == '4',
   Typ=groesse,
   Ordnung=ab,!
   ;
   write('Falsche Eingabe')
   ).



/* ********* ENDE ********* */
```

Literatur

Abelson, R.P. (1968). Simulation of Social Behavior. In G. Lindzey/E. Aronson (eds.), *The Handbook of Social Psychology* Vol. 2 (2.Aufl.). Reading, Mass: Addison-Wesley, S.274-356.

Abelson, R.P. (1973). The Structure of a Belief System. In R.C. Schank/K.M. Colby (eds.), *Computer Models of Thought and Language*. San Francisco: Freeman.

Abelson, R.P./Rosenberg, M.J. (1958). Symbolic Psycho-Logic: A Model of Attitudinal Cognition. *Behavioral Science*, 3, S.1-13.

Allport, D.A. (1980). Patterns and Actions. In G. Claxton (ed.), *Cognitive Psychology: New Directions*. London: Routledge & Degan, S.26-64.

Anderson, B. (1979). Cognitive Balance Theory and Social Network Analysis: Remarks on Some Fundamental Theoretical Matters. In P.W.Holland/S.Leinhardt (eds.), S.453-469.

Anderson, B. (1989). On Artificial Intelligence and Theory Construction in Sociology. *Journal of Mathematical Sociology*, 14, S.209-216.

Apter, M.J. (1971). *The Computer Simulation of Behaviour*. New York: Harper & Row.

Bainbridge, W.S. (1987). *Sociology Laboratory. Computer Simulations for Learning Sociology*. Belmont: Wadsworth Publishing.

Balzer, W. (1982). *Empirische Theorien: Modelle - Strukturen - Beispiele. Die Grundzüge der modernen Wissenschaftstheorie*. Braunschweig: Vieweg.

Balzer, W. (1985). *Theorie und Messung*. Berlin: Springer.

Balzer, W./Moulines, C.U./Sneed, J.D. (1987). *An Architectonic for Science. The Structuralist Program*. Dordrecht: Reidel.

Banerjee, S. (1986). Reproduction of Social Structure. An Artificial Intelligence Model. *Journal of Conflict Resolution*, 30, 2, S. 221-252.

Barr, A./Feigenbaum, E.A. (1981). *The Handbook of Artificial Intelligence* Vol. 1, Los Altos: Kaufman.

Bauer, K. (1989). *Programmieren in Prolog für Wissenschaftstheoretiker*. Handout zum Proseminar an der Universität München, Seminar für Philosophie, Logik und Wissenschaftstheorie. Unveröffentlichtes Manuskript.

Belli, F. (1986). *Einführung in die logische Programmierung mit Prolog*. Mannheim: Bibliographisches Institut.

Benfer, R.A./Brent, E.E./Furbee, L. (1991). *Expert Systems*. Newbury Park CA: Sage.

Berger, J./Cohen, B.P./Snell, J.L./Zelditch, M. (1962). *Types of Formalization in Small-Group Research*. Boston.

Bibel, W./Siekmann, J.H. (1994). Informatik und Intellektik als zukünftiges Gespann. *KI* 1, S.16-22.

Boden, M. (1984). Artificial Intelligence and Social Forecasting. *Journal of Mathematical Sociology*, 9, S.341-356.

Boden, M. (1987). *Artificial Intelligence and Natural Man* (2.Aufl.). New York: Basic Books.

Boden, M. (1988). *Computer Models of Mind*. Cambridge: University Press.

Bossel, H. (1992). *Modellbildung und Simulation. Konzepte, Verfahren und Modelle zum Verhalten dynamischer Systeme*. Braunschweig: Vieweg.

Bratko, I. (1987). *Prolog. Programmierung für Künstliche Intelligenz*. Bonn: Addison-Wesley (zuerst 1986: Prolog Programming for Artificial Intelligence. Reading, Mass: Addison-Wesley).

Brent, E.E. (1986). Knowledged-Based Systems: A Qualitative Formalism. *Qualitative Sociology*, 9, 3, S.256-282.

Bron, C./Kerbosch, J. (1973). Algorithm 457 (H). Finding all Cliques of an Undirected Graph. *Communications of the ACM*, 16, S.575-577.

Bunge, M. (1983). *Epistemologie. Aktuelle Fragen der Wissenschaftstheorie*. Mannheim: Bibliographisches Institut.

Burnstein, E. (1967). Sources of Cognitive Bias in the Representation of Simple Social Structures: Balance, Minimal Chance, Positivity, Reciprocity, and the Respondents own Attitude. *Journal of Personality and Social Psychology*, 7, S.36-48.

Carnap, R. (1986). *Einführung in die Philosophie der Naturwissenschaften*. Frankfurt a.M.: Ullstein (zuerst 1966: Philosophical Foundations of Physics. New York: Basic Books).

Cartwright, D./Harary, F. (1956). Structural Balance: A Generalisation of Heider's Theory. *Psychological Review*, 63, S. 277-293 (wieder abgedruckt in: S.Leinhardt (ed.) 1977, S.9-25).

Cartwright. D./Harary, F. (1979). Balance and Clusterability: An Overview. In P.W.Holland/S.Leinhardt (eds.), S.25-50.

Claus, V. (1986). *Informatik*. Mannheim: Bibliographisches Institut.

Clocksin, W.F./Mellish, C.S. (1987). *Programming in Prolog* (3. Aufl.). Berlin: Springer.

Cohen, P.S. (1980). Soziologische Theorie. In J.Speck (ed.), *Handbuch wissenschaftstheoretischer Begriffe* Band 3. Göttingen: Vandenhoeck und Ruprecht, S.592-597.

Colby, K.M. (1973). Simulations of Belief Systems. In R.Schank/K.M.Colby (eds.), *Computer Models of Thought and Language*. San Francisco: Freeman, S. 251-286.

Colby, K.M. (1975). *Artificial Paranoia: A Computer Simulation of Paranoid Processes*. New York: Pergamon.

Collani von, G. (1987). Anmerkungen zu Rechenverfahren für die Ermittlung von Cliquen. Eine Replik auf Freeman & Libhart. *Zeitschrift für Sozialpsychologie*, 18, S.131-133.

Davis, J.A. (1967). Clustering and Structural Balance in Graphs. *Human Relations*, 20, S.181-187 (wieder abgedruckt in: S.Leinhardt (ed.) 1977, S.27-33).

Davis, J.A. (1970). Clustering and Hierarchy in Interpersonal Relations: Testing two Graph Theoretical Models on 742 Sociogramms. *American Sociological Review*, 35, S.27-33.

Davis, J.A./Leinhardt, S. (1972). The Structure of Positive Interpersonal Relations in Small Groups. In J.Berger (ed.), *Sociological Theories in Progress* Vol. 2. Boston: Houghton-Mifflin, S.218-253.

Dawson, R.E. (1962). Simulation in the Social Sciences. In H.Guetzkow (ed.), *Simulation in Social Science*. Englewood Cliffs, N.J.: Prentice-Hall, S.1-15.

Dennett, D.C. (1978). *Brainstorms. Philosophical Essays on Mind and Psychology*. Hassocks: Harvester.

Deutsch, M./Krauss, R.M. (1976). *Theorien der Sozialpsychologie*. Frankfurt a.M.: Fachbuchhandlung Psychologie.

Dollase, R. (1973). *Soziometrische Techniken*. Weinheim: Beltz.

Doran, J. (1985). The Computational Approach to Knowledge, Communication and Structure in Multi-Actor-Systems. In G.N.Gilbert/C.Heath (eds.), S.160-171.

Dörner, D. (1984). Modellbildung und Simulation. In E.Roth (ed.), *Sozialwissenschaftliche Methoden*. München: Oldenbourg, S. 337-350.

Esser, H./Troitzsch, K.G. (1991b). Einleitung: Probleme der Modellierung sozialer Prozesse. In H.Esser/K.G.Troitzsch (eds.) (1991a), S.13-25.

Esser, H./Troitzsch, K.G. (eds.) (1991a). *Modellierung sozialer Prozesse*. Bonn: Informationszentrum Sozialwissenschaften.

Faulbaum, F. (1986). *Very Soft Modeling*. ZUMA-Arbeitsbericht Nr. 86/04.

Faulbaum, F. (1991). Von der Variablensoziologie zur empirischen Evaluation von Handlungsparadigmen. In H.Esser/K.G.Troitzsch (eds.) (1991a), S.111-138.

Feigenbaum, E.A. (1963). The Simulation of Verbal Learning Behavior. In E.A.Feigenbaum/J.Feldman (eds.), *Computers and Thought*. New York: McGraw-Hill, S.299-325.

Festinger, L. (1978). *Theorie der kognitiven Dissonanz*. Bern: Huber (zuerst 1957: A Theory of Cognitive Dissonance. Evanstone, Ill.: Row Peterson).

Fodor, J.A. (1968). *Psychological Explanation*. New York: Random House.

Fodor, J.A. (1980). Methodological Solipsism Considered as a Research Strategy in Cognitive Psychology. *The Behavioral and Brain Sciences*, 3, S.63-110.

Forrester, J.W. (1971). *World Dynamics*. Cambridge, Mass: Wright-Allen Press.

Freeman, L.C./Libhart, D.L. (1986). Eine Anmerkung zu Rattingers Algorithmus zur Ermittlung von Cliquen. *Zeitschrift für Sozialpsychologie*, 17, S.119-121.

Frijda, N.H. (1967), Problems of Computer Simulation. In J.M.Dutton/W.H.Starbuck (eds.), *Computer Simulation of Human Behavior*. New York: Wiley, S.610-618.

Garson, G.D. (1987). The Role of Inductive Expert Systems Generators in the Social Science Research Process. *Social Science Microcomputer Review*, 5, 1, S.11-21.

Ghezzi, C./Jazayeri, M. (1989). *Konzepte der Programmiersprachen: begriffliche Grundlagen, Analyse und Bewertung*. München: Oldenbourg.

Giesen, B./Schmid, M. (1977). *Basale Soziologie: Wissenschaftstheorie*. Opladen: Westdeutscher Verlag.

Gilbert, G.N./Heath, C. (eds.) (1985). *Social Action and Artificial Intelligence*. Aldershot: Gower.

Glory, J.C./Masuch, M./Marx, M. (1990). Formalizing Organizational Theory: A Knowledge-Based Approach. In M. Masuch (ed.), *Organization, Management, and Expert Systems. Models of Automated Reasoning*. Berlin: de Gruyter, S. 79-104.

GMD Gesellschaft für Mathematik und Datenverarbeitung (1987). *Die Sprache Prolog*. GMD F2G2.

Goldschlager, L./Lister, A. (1990). *Informatik. Eine moderne Einführung* (3.Aufl.). München: Hanser.

Gollob, H.F. (1974). The subject-verb-object approach to social cognition. *Psychological Review*, 81, S.286-321.

Görz, G. (1988). *Strukturanalyse natürlicher Sprache*. Bonn: Addison-Wesley.

Götschl, J. (1980). Theorie. In J.Speck (ed.), *Handbuch wissenschaftstheoretischer Begriffe* Band 3. Göttingen: Vandenhoeck & Ruprecht, S.636-646.

Granovetter, M.S. (1973). The Strength of Weak Ties. *American Journal of Sociology*, 78, S.1360-1380 (wieder abgedruckt in: S.Leinhardt (ed.) 1977, S.347-367).

Granovetter, M.S. (1979). The Theory-Gap in Social Network Analysis. In P.W.Holland/S.Leinhardt, S.501-518.

Granovetter, M.S. (1982). The Strength of Weak Ties. A Network Theory Revisited. In P.V.Marsden/N.Lin (eds.), *Social Structure and Network Analysis*. Beverly Hills: Sage, S.105-130.

Gullahorn, J.T./Gullahorn, J.E. (1963). A Computer Model of Elementary Social Behavior. *Behavioral Science*, 8, S.354-362 (deutsch in: R.Mayntz (ed.), *Formalisierte Modelle in der Soziologie*. Neuwied/Berlin: Luchterhand, S. 233-248).

Gullahorn, J.T./Gullahorn, J.E. (1965). The Computer as a Tool for Theory Development. In D.Hymes (ed.), *The Use of Computers in Anthropology*. London: Mouton, S.427-448.

Haas, J. (1990). Treatment of Uncertainty in Social-Science Expert Systems. In H.Czap/W.Nedobity (eds.), *Terminology and Knowledge Engineering* Vol.1. Frankfurt a.M.: Indeks, S.62-76.

Hallinan, M.T. (1974). A Structural Model of Sentiment Relations. *American Journal of Sociology*, 80, 2, S.364-378.

Hallinan, M.T./Felmlee, D. (1975). An Analysis of Intransitivity in Sociometric Data. *Sociometry*, 38, S.195-212.

Halmos, P.R. (1976). *Naive Mengenlehre* (4.Aufl.). Göttingen: Vandenhoeck & Ruprecht (zuerst 1968: Naive Set Theory. Princeton: van Nostrand).

Harary, F. (1953). On Local Balance and N-Balance in Signed Graphs. *Mich. Mathematical Journal*, 2, S.143-146.

Harary, F. (1969). *Graph Theory*. Reading, Mass: Addison-Wesley.

Harary, F./Norman, R.Z./Cartwright, D. (1965). *Structural Models: An Introduction to the Theory of Directed Graphs*. New York: Wiley.

Harary, F./Ross, I.C. (1957). A Procedure for Clique Detection Using the Group Matrix. *Sociometry*, 20, S.205-215.

Harbordt, S. (1974). *Computersimulation in den Sozialwissenschaften* 2 Bände, Reinbek bei Hamburg: Rowohlt.

Harmon, P./King, D. (1986). *Expertensysteme in der Praxis. Perspektiven, Werkzeuge, Erfahrungen*. München: Oldenbourg.

Harmon, P./Maus, R./Morrissey, W. (1989). *Expertensysteme: Werkzeuge und Anwendungen*. München: Oldenbourg.

Haugeland, J. (ed.) (1981). *Mind Design. Philosophy, Psychology, Artificial Intelligence*. Cambridge, Mass: MIT Press.

Hayes-Roth, F./Waterman, D.A./Lenat, D.B. (1983). *Building Expert Systems*. Reading, Mass: Addison-Wesley.

Heider, F. (1946). Attitudes and Cognitive Organization. *Journal of Psychology*, 21, S.107-112 (wieder abgedruckt in: S.Leinhardt (ed.) 1977, S.3-8).

Heider, F. (1977). *Psychologie der interpersonalen Beziehungen*. Stuttgart: Klett (zuerst 1958: The Psychology of Interpersonal Relations. New York: Wiley).

Heider, F. (1979). On Balance and Attribution. In P.W.Holland/S.Leinhardt, S.11-23.

Heyer, G. (1988). Geist, Verstehen und Verantwortung. Philosophische Grundlagen der Künstlichen Intelligenz Teil 1. *KI*, 1, S.36-40.

Heyer, G./Krems, J./Görz, G. (eds.) (1988). *Wissensarten und ihre Darstellung. Beiträge aus Philosophie, Psychologie, Informatik und Linguistik*. Berlin: Springer.

Hogeweg, P./Hesper, B. (1985). Socioinformatic Processes: MIRROR Modelling Methodology. *Journal of Theoretical Biology*, 113, S.311-330.

Holland, P.W./Leinhardt, S. (1970). A Method for Detecting Structure in Sociometric Data. *American Journal of Sociology*, 70, S.492-513 (wieder abgedruckt in: S.Leinhardt 1977, S.411-432).

Holland, P.W./Leinhardt, S. (1971). Transitivity in Structural Models of Small Groups. *Comparative Group Studies* 2, S.107-124 (wieder abgedruckt in: S.Leinhardt 1977 (ed.), S.49-66).

Holland, P.W./Leinhardt, S. (1975). Structural Sociometry. Papier präsentiert auf dem *Advanced Research Symposium on Social Networks*, Mathematical Social Science Board, Hanover, New Hampshire (September).

Holland, P.W./Leinhardt, S. (eds.) (1979). *Perspectives on Social Network Research*. New York: Academic Press.

Homans, G.C. (1950). *The Human Group*. New York: Harcourt.

Homans, G.C. (1961). *Social Behaviour: Its Elementary Forms*. New York: Harcourt.

Hopcroft, J.E. (1984). Turingmaschinen. *Spektrum der Wissenschaft*, 6, S.34-49.

Hughes, P./Brecht, G. (1978). *Die Scheinwelt des Paradoxons*. Braunschweig: Vieweg.

Hummell, H.J. (1972a). Zur Problematik der Ableitung in sozialwissenschaftlichen Aussagensystemen. Ein Plädoyer für Formalisierung (Teil 1). *Zeitschrift für Soziologie*, 1, S.31-46.

Hummell, H.J. (1972b). Zur Problematik der Ableitung in sozialwissenschaftlichen Aussagensystemen. Ein Plädoyer für Formalisierung (Teil 2). *Zeitschrift für Soziologie*, 2, S.118-138.

Hummell, H.J./Sodeur, W. (1984). Interpersonelle Beziehungen und Netzstruktur. *Kölner Zeitschrift für Soziologie und Sozialpsychologie*, 36, S.494-510.

Hummell, H.J./Sodeur, W. (1987). Triaden- und Triplettzensus als Mittel der Strukturbeschreibung. In J. van Koolwijk/M.Wieken-Mayser (eds.), S.129-161.

Irle, M. (1975). *Lehrbuch der Sozialpsychologie*. Göttingen: Hogrefe.

Johnsen, E.C. (1985). Network Macrostructure Models for the Davis-Leinhardt-Set of Empirical Sociomatrices. *Social Networks*, 5, S.203-224.

Jordan, N. (1953). Behavioral Forces that are a Function of Attitude and Cognitive Organization. *Human Relations*, 6, S.273-287.

Jungnickel, D. (1990). *Graphen, Netzwerke und Algorithmen* (2. Aufl.). Mannheim: Bibliographisches Institut.

Kappelhoff, P. (1987), Cliquenanalyse. Die Bestimmung von intern verbundenen Teilgruppen in sozialen Netzwerken. In J.van Koolwijk/M.Wieken-Mayser (eds.), S.39-63.

Kleine-Büning, K./Schmitgen, S. (1986). *Prolog. Grundlagen und Anwendungen*. Stuttgart: Teubner.

Kleinknecht, R./Wüst, E. (1976). *Lehrbuch der elementaren Logik* 2 Bände. München: dtv.

Knödel, W. (1968). Algorithmus 8. Bestimmung aller maximalen, vollständigen Teilgraphen eines Graphen G nach Stoffers. *Computing*, 3, S.239-240.

Knoke, D./Kuklinski, J.H. (1982). *Network Analysis*. Beverly Hills/London: Sage.

Kobsa, A. (1982). On Regarding AI Programs as Theories. In R.Trappl (ed.), *Cybernetics and Systems Research*. Amsterdam: North-Holland, S.933-935.

Kobsa, A. (1984). What is Explained by AI Models. *Communication & Cognition*, 17, 2/3, S.49-65.

Kobsa, A. (1986). Artificial Intelligence und Kognitive Psychologie. In J.Retti et al (eds.), *Artificial Intelligence. Eine Einführung* (2.Aufl.). Stuttgart: Teubner, S.105-130.

Koolwijk van, J./Wieken-Mayser, M. (eds.) (1987). *Techniken der empirischen Sozialforschung* Bd. 1. Methoden der Netzwerkanalyse. München: Oldenbourg.

Koukkanen, M. (1992). The Continuity Problem of Scientific Theories. An Example of Social-Psychological Balance Theorizing. In H.Westmeyer (ed.), S.211-248.

Kowalski, R. (1988). *Logic for Problem Solving* (9. Aufl.). New York: North Holland.

Krämer, S. (1988). *Symbolische Maschinen. Die Idee der Formalisierung im geschichtlichen Abriß*. Darmstadt: Wissenschaftliche Buchgesellschaft.

Kreutz, H./Bacher, J. (eds.) (1991). *Disziplin und Kreativität. Sozialwissenschaftliche Computersimulation: theoretische Experimente und praktische Anwendung*. Opladen: Leske + Budrich.

Kreutzer, W. (1986). *System Simulation. Programming Styles and Languages*. Sidney: Addison-Wesley.

Kuhn, T.S. (1991). *Die Struktur wissenschaftlicher Revolutionen* (11.Aufl.). Frankfurt a.M.: Suhrkamp (zuerst 1962: The Structure of Scientific Revolutions. Chicago: University Press).

Lakatos, I. (1982). *Die Methodologie der wissenschaftlichen Forschungsprogramme*. Braunschweig: Vieweg.

Landy, D./Aronson, E. (1969). The Influence of the Character of the Criminal and his Victim on the Decisions of Simulated Jurors. *Journal of Experimental Social Psychology*, 5, S.141-152.

Langley, P. (1979). Rediscovering Physics with BACON.3. *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* Vol.1, Tokyo, S.505-507.

Langley, P./Simon, H.A./Bradshaw, G.L./Zytkow, J.M. (1987). *Scientific Discovery: Computational Explorations of the Creative Processes*. Cambridge, Mass: MIT Press.

Leinhardt, S. (ed.) (1977). *Social Networks. A Developing Paradigm*. New York: Academic Press.

Lerner, M.J./Simmons, C.H. (1966). Observer's Reaction to the "Innocent Victim": Compasion of Rejection? *Journal of Personality and Social Psychology*, 4, S.203-210.

Lindenberg, S. (1971). Simulation und Theoriebildung. In H.Albert (ed.), *Sozialtheorie und soziale Praxis*. Meisenheim: Hain, S.78-113.

Lischka, C./Diederich, J. (1987). Gegenstand und Methode der Kognitionswissenschaft. *GMD-Spiegel*, 2/3, S.21-32.

Lloyd, J.W. (1993). *Foundations of Logic Programming* (2. Aufl.). Berlin: Springer.

Loeck, G. (1986). Ist Simulation Erklärung? Cognitive Science - wissenschaftstheoretisch betrachtet. *Zeitschrift für allgemeine Wissenschaftstheorie*, 17, S.14-39.

Luce, R.D./Perry, A.D. (1949). A Method of Matrix Analysis of Group Structure. *Psychometrika*, 14, S.95-116.

Ludewig, J. (1985). *Sprachen für die Programmierung. Eine Übersicht*. Mannheim: Bibliographisches Institut.

Manhart, K. (1987). *Analyse sozialer Strukturen mit Prolog*. Unveröffentlichtes Manuskript.

Manhart, K. (1988). Prolog und Logik. Der Beweismechanismus von Prolog und seine logischen Grundlagen. Teil 1: *mc* 2/88, S.41-43; Teil 2: *mc* 3/88, S.87-89.

Manhart, K. (1989). Können AI-Programme als Theorien betrachtet werden? Ein wissenschaftsphilosophischer Beitrag. In J.Retti/K.Leidlmeier (eds.), *Proceedings 5. Österreichische AI-Tagung*. Berlin: Springer, S.246-358.

Manhart, K. (1991). KI-Modellierung in den Sozialwissenschaften. *KI*, 2, S.32-40.

Markus, H./Zajonc, R.B. (1985). The Cognitive Perspective in Social Psychology. In G. Lindzey/E. Aronson (eds.), *The Handbook of Social Psychology* Vol. 1 (3.Aufl.). New York: Random, S.137-230.

Martial von, F. (1992). Einführung in die Verteilte Künstliche Intelligenz. *KI*, 1, S.6-11.

Mayntz, R. (1967). Modellkonstruktion: Ansatz, Typen und Zweck. In R.Mayntz (ed.), *Formalisierte Modelle in der Soziologie*. Neuwied/Berlin: Luchterhand.

Mayntz, R./Holm, K./Hübner, P. (1974). *Einführung in die Methoden der empirischen Soziologie* (4.Aufl.). Opladen: Westdeutscher Verlag.

McPhee, W.N. (1960). Computer Models and Social Theory: An Introduction, Paper Presented at the Annual Meetings of the American Sociological Association, New York.

Meadows, D./Meadows, D./Jahn, E./Milling, P. (1972). *Die Grenzen des Wachstums. Bericht des Club of Rome zur Lage der Menschheit*. Stuttgart: Deutsche Verlagsanstalt.

Merritt, D. (1989). *Building Expert Systems in Prolog*. Berlin: Springer.

Mohazab, F./Feger, H. (1985). An Extension of Heiderian Balance Theory for Quantified Data. *European Journal of Social Psychology*, 15, S.147-165.

Möhring, M. (1990). *Mimose. Eine funktionale Sprache zur Beschreibung und Simulation individuellen Verhaltens in interagierenden Populationen*. Dissertation, Universität Koblenz-Landau.

Müller, J. (ed.) (1993). *Verteilte Künstliche Intelligenz. Methoden und Anwendungen*. Mannheim: Bibliographisches Institut

Mulligan, G.D./Corneil, D.G. (1972). Corrections to Bierstone' s Algorithm for Generating Cliques. *Journal of the ACM*, 19, S.244-247.

Newcomb, T.M. (1953). An Approach to the Study of Communicative Acts. *Psychological Review*, 60, S.393-404.

Newcomb, T.M. (1961). *The Acquaintance Process*. New York: Holt, Rinehart and Winston.

Newell, A./Simon, H.A. (1961). GPS - A Program that Simulates Human Thought. In H.Billing (ed.), *Lernende Automaten*. München: Oldenbourg, S.109-124.

Newell, A./Simon, H.A. (1971). Simulation of Human Thought. In J.Dutton/W.H.Starbuck (eds.), *Computer Simulation of Human Behavior*. New York: Wiley, S.150-169.

Newell, A./Simon, H.A. (1972). *Human Problem Solving*. Englewood Cliffs, N.J.: Prentice-Hall.

Opp, K.D. (1976). *Methodologie der Sozialwissenschaften. Einführung in Probleme ihrer Theoriebildung*. Hamburg: Rowohlt.

Opp, K.D. (1984). Balance Theory: Progress and Stagnation of a Social Psychological Theory. *Philosophy of the Sociological Sciences*, 14, 1, S.27-49.

Opwis, K. (1992). *Kognitive Modellierung. Zur Verwendung wissensbasierter Systeme in der psychologischen Theoriebildung*. Bern: Huber.

Osgood, C.E./Tannenbaum, P.H. (1955). The Principle of Congruity in the Prediction of Attitude Change. *Psychological Review*, 62, S.42-55.

Ostrom, T.M. (1988). Computer Simulation: The Third Symbol System. *Journal of Experimental Social Psychology*, 24, S.381-392.

Péli, G./Bruggeman, J./Masuch, M./Nuallàin, B. (1992). A Logical Approach to Organizational Ecology: Formalizing the Inertia-Fragment in First-Order Logic. CCSOM Reprint 92-74 (Center for Computer Science in Organization and Managment).

Popper, K.R. (1987). *Das Elend des Historizismus* (6.Aufl.). Tübingen: Mohr (zuerst 1960: The Poverty of Historicism. London: Routledge & Kegan).

Popper, K.R. (1972). Zur Logik der Sozialwissenschaften. In T.W.Adorno et al (eds.), *Der Positivismusstreit in der deutschen Soziologie*. Neuwied/Berlin: Luchterhand. S.103-123.

Popper, K.R. (1982). *Logik der Forschung* (7.Aufl.). Tübingen: Mohr (zuerst 1934: Logik der Forschung. Wien: Springer).

Prim, R./Tilmann, H. (1989). *Grundlagen einer kritisch-rationalen Sozialwissenschaft. Studienbuch zur Wissenschaftstheorie* (6.Aufl.). Heidelberg: Quelle und Meier.

Puppe, F. (1988). *Einführung in Expertensysteme*. Berlin: Springer.

Pylyshyn, Z.W. (1980). Computation and Cognition: Issues in the Foundations of Cognitive Science. *The Behavioral and Brain Sciences*, 3, S.111-139.

Pylyshyn, Z.W. (1984). *Computation and Cognition. Toward a Foundation of Cognitive Science*. Cambridge, Mass: MIT Press.

Rapoport, A. (1980). *Mathematische Methoden in den Sozialwissenschaften*. Heidelberg: Physica.

Rattinger, H. (1973). Eine einfache Methode und ein FORTRAN-Programm zur Ermittlung von Cliquen. *Zeitschrift für Sozialpsychologie*, 4, S.5-14.

Rechenberg, P. (1991). *Was ist Informatik? Eine allgemeinverständliche Einführung*. München: Hanser.

Reimer, U. (1991). *Einführung in die Wissensrepräsentation*. Stuttgart: Teubner.

Reischuk, K.R. (1990). *Einführung in die Komplexitätstheorie*. Stuttgart: Teubner.

Rich, E. (1988). *KI-Einführung und Anwendungen*. Hamburg: McGraw-Hill (zuerst 1983: Artificial Intelligence. New York: McGraw-Hill).

Robinson, J.A. (1965). A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12, 1, S.23-41.

Rosenberg. M.J./Abelson, R.P. (1960). An Analysis of Cognitive Balancing. In M.J.Rosenberg/C.I.Hovland/W.J.McGuire/R.P.Abelson/J.W.Brehm (eds.), *Attitude Organization and Change*. New Haven, Connect.: Yale University Press.

Savory, S.E. (1988). *Grundlagen von Expertensystemen*. München: Oldenbourg.

Schelling, T.C. (1978). *Micromotives and Macrobehavior*. New York: Norton.

Schenk, M. (1984). *Soziale Netzwerke und Kommunikation*. Tübingen: Mohr.

Schnell, R. (1990). Computersimulation und Theoriebildung in den Sozialwissenschaften. *Kölner Zeitschrift für Soziologie und Sozialpsychologie*, 42, 1, S.109-128.

Schnell, R./Hill, P.B./Esser, E. (1993). *Methoden der empirischen Sozialforschung* (4.Aufl.). München: Oldenbourg.

Schnupp, P. (1986). *Prolog. Einführung in die Programmierpraxis*. München: Hanser.

Schnupp, P./Nguyen Huu, C.T. (1987). *Expertensystem-Praktikum*. Berlin: Springer.

Shortliffe, E.H. (1976). *Computer-Based Medical Consultations: Mycin*. New York: Elsevier.

Simon, H.A. (1957). *Models of Man, Social and Rational. Mathematical Essays on Rational Human Behavior in a Social Setting*. New York: Wiley.

Simon, H.A. (1981). *The Sciences of the Artificial* (2.Aufl.). Cambridge, Mass: MIT Press.

Simon, H.A./Newell, A. (1956). Models: Their Uses and Limitations. In L.D.White (ed.), *The State of the Social Sciences*. Chicago: University Press. S.66-83.

Simon, T.W. (1979). Philosophical Objections to Programs as Theories. In M.Ringle (ed.), *Philosophical Perspectives in Artificial Intelligence*. Hassocks: Harvester, S.225-242.

Slezak, P. (1989). Scientific Discovery by Computer as Empirical Refutation of the Strong Programme. *Social Studies of Science*, 19, S.563-600.

Sneed, J.D. (1971). *The Logical Structure of Mathematical Physics*. Dordrecht: Reidel (2. Auflage 1979).

Stahlberg, D./Frey, D. (1987). Konsistenztheorien. In D.Frey/S.Greif (eds.), *Sozialpsychologie* (2.Aufl.). München: Psychologie Verlags Union, S.214-221.

Stegmüller, W. (1973). *Personelle und Statistische Wahrscheinlichkeit* (Probleme und Resultate der Wissenschaftstheorie und Analytischen Philosophie. Band IV, Studienausgabe). Berlin: Springer.

Stegmüller, W. (1978). *Hauptströmungen der Gegenwartsphilosophie. Eine kritische Einführung* Band I (6. Aufl.). Stuttgart: Kröner.

Stegmüller, W. (1979). *Hauptströmungen der Gegenwartsphilosophie. Eine kritische Einführung* Band II (6. Aufl.). Stuttgart: Kröner.

Stegmüller, W. (1980). *Neue Wege der Wissenschaftsphilosophie*. Berlin: Springer.

Stegmüller, W. (1983). *Erklärung - Begründung - Kausalität* (Probleme und Resultate der Wissenschaftstheorie und Analytischen Philosophie, Band I. 2.Aufl.). Berlin: Springer.

Stegmüller, W. (1986). *Theorie und Erfahrung: Dritter Teilband. Die Entwicklung des neueren Strukturalismus seit 1973.* (Probleme und Resultate der Wissenschaftstheorie und Analytischen Philosophie, Band II). Berlin: Springer.

Stegmüller, W./Varga von Kibéd, M. (1984). *Strukturtypen der Logik*. Teil A. (Probleme und Resultate der Wissenschaftstheorie und Analytischen Philosophie, Band III, Studienausgabe). Berlin: Springer.

Stephan, E. (1990). *Zur logischen Struktur psychologischer Theorien*, Berlin: Springer.

Sterling, L./Shapiro, E. (1988). *Prolog. Fortgeschrittene Programmiertechniken*. Bonn: Addison-Wesley.

Stillings, N.A. et al. (1987). *Cognitive Science. An Introduction*. Cambridge, Mass: MIT Press.

Stoyan, H. (1988). *Programmiermethoden der Künstlichen Intelligenz* Band 1. Berlin: Springer.

Sukale, M. (1971). Zur Axiomatisierung der Balancetheorie. Eine wissenschaftstheoretische Fallstudie. *Zeitschrift für Sozialpsychologie*, 2, S.40-57.

Suppes, P. (1983). Warum Formalisierung in der Wissenschaft erwünscht ist. In W.Balzer/M.Heidelberger, S.24-39 (zuerst 1968: The Desirability of Formalization in Science. Journal of Philosophy, 65, S.651-664).

Sylvan, D./Glassner, B. (1985). *A Rationalist Methodology for the Social Sciences*. New York: Blackwell.

Tanimoto, S.L. (1990). *KI: Die Grundlagen*. München: Oldenbourg.

Tarski, A. (1977). *Einführung in die mathematische Logik* (5.Aufl.). Göttingen: Vandenhoeck & Ruprecht.

Thorson, S.J./Sylvan, D.A. (1982). Counterfactuals and the Cuban Missile Crisis. *International Studies Quarterly*, 26, 4, S.539-571.

Troitzsch, K. (1990). *Modellbildung und Simulation in den Sozialwissenschaften*. Opladen: Westdeutscher Verlag.

Turing, A.M. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2, 42, S.230-265.

Tutte, W.T. (1984). *Graph Theory*. Reading, Mass: Addison-Wesley.

Waltz, D.L. (1982). Künstliche Intelligenz. *Spektrum der Wissenschaft*, 12, S.68-87.

Waterman, D.A. (1986). *A Guide to Expert Systems*. Reading, Mass: Addison-Wesley.

Weizenbaum, J. (1978). *Die Macht der Computer und die Ohnmacht der Vernunft*. Frankfurt a.M.: Suhrkamp (zuerst 1976: Computer Power and Human Reason. From Judgement to Calculation. San Francisco: Freeman).

Wessels, M.G. (1984). *Kognitive Psychologie*. New York: Harper & Row.

West, S.G./Wicklund, R.D. (1985). *Einführung in sozialpsychologisches Denken*. Weinheim: Beltz.

Westmeyer, H. (ed.) (1992), *The Structuralist Program in Psychology*. Bern: Huber.

Westermann, R. (1987). *Strukturalistische Theorienkonzeption und empirische Forschung in der Psychologie. Eine Fallstudie*. Berlin: Springer.

Wiswede, G. (1988). Umrisse einer integrativen Lerntheorie sozialen Verhaltens. *Zeitschrift für Sozialpsychologie*, 19, S.17-30.

Winston, P.H. (1984). *Artificial Intelligence* (2.Aufl.). Reading, Mass: Addison-Wesley.

Winston, P.H. (1987). *Künstliche Intelligenz*. Bonn: Addison-Wesley.

Winston, P.H./Horn, B.K.P. (1987). *Lisp*. Bonn: Addison-Wesley (zuerst 1984: Lisp. Reading, Mass: Addison-Wesley).

Witte, E.H. (1989). *Sozialpsychologie. Ein Lehrbuch*. München: Psychologie Verlags Union.

Zeigler, B.P. (1976). *Theory of Modeling and Simulation*. New York: Wiley.

Ziegler, R. (1972). *Theorie und Modell. Der Beitrag der Formalisierung zur soziologischen Theoriebildung*. München: Oldenbourg.

Namenregister

Sachregister